

Beschreibung

Die vorliegende Erfindung nimmt bezug auf die US-Patentanmeldung mit dem Titel: "System und Verfahren für phasensynchrones Taktieren mit flexibler Frequenz und phasensynchrone Nachrichtenübertragung" bzw. "System and Method for Phase-Synchronous, Flexible Frequency Clocking and Messaging".

Die vorliegende Erfindung betrifft im allgemeinen die Architektur von Computern und insbesondere Systeme und Verfahren zum rekonfigurierbaren bzw. umstrukturierbaren Rechnen. Noch genauer handelt es sich bei der vorliegenden Erfindung um ein System und ein Verfahren zum skalierbaren, parallelen, dynamisch rekonfigurierbaren Rechnen.

Die Evolution der Computerarchitektur wird von dem Bedürfnis nach immer größerer Rechenleistung angetrieben. Eine schnelle und genaue Lösung verschiedener Arten von Rechenproblemen oder numerischen Problemen erfordert typischerweise verschiedene Arten von Rechen-Systemelementen bzw. Rechen-Resourcen. Ist ein bestimmter Problembereich gegeben, so kann die Rechenleistung durch die Verwendung von Rechen-Systemelementen erhöht werden, die speziell für die betrachteten Problemtypen architektonisch ausgelegt bzw. strukturiert wurden. Zum Beispiel kann die Verwendung einer Hardware zur digitalen Signalverarbeitung (DSP) bzw. eine "Digital Signal Processing"-Hardware in Verbindung mit einem Allzweck-Computer signifikant bestimmte Arten der Signalverarbeitungsleitung bzw. der Signalverarbeitungswirkung erhöhen. Für den Fall, daß ein Computer selbst speziell für die betrachteten Problemtypen architektonisch ausgelegt wurde, wird die Berechnungsleistungsfähigkeit weiter erhöht werden oder möglicherweise sogar in bezug auf die verfügbaren Rechen-Systemelemente bzw. Rechen-Resourcen für diese bestimmten Problemtypen optimiert werden. Gegenwärtige parallele oder massiv parallele Computer, die eine hohe Leistungsfähigkeit für bestimmte Typen von Problemen der Ordnung n^2 bzw. von $O(n^2)$ oder größerer Komplexität anbieten, stellen Beispiele für diesen Fall dar.

Das Bedürfnis nach größerer Rechenleistung muß gegenüber dem Bedürfnis nach der Minimierung der Systemkosten und dem Bedürfnis nach der Maximierung der Systemleistungsfähigkeit in einem breitestmöglichen Bereich sowohl der täglichen bzw. aktuellen als auch möglichen zukünftigen Anwendungen abgewogen werden. Im allgemeinen beeinträchtigt das Einbringen von Rechen-Systemelementen bzw. Rechen-Resourcen, die auf eine begrenzte Anzahl von Problemtypen ausgerichtet sind, in ein Computersystem nachteilig die Systemkosten, weil eine spezialisierte Hardware typischerweise teurer ist als eine Allzweck-Hardware. Die Gestaltung und die Erzeugung eines ganzen Computers für einen bestimmten Zweck kann so teuer werden, daß es sich verbietet, und zwar sowohl bezüglich der Erstellungszeit als auch der Hardwarekosten. Die Verwendung spezialisierter Hardware, um die Rechenleistung zu erhöhen, kann wenig positives bezüglich der Leistungsfähigkeit anbieten, da die Bedürfnisse bezüglich der Berechnungen sich ändern. Im Stand der Technik wurden, da sich die Erfordernisse bezüglich der Berechnung geändert haben, neue Typen spezialisierter Hardware oder neue Systeme für bestimmte Zwecke gestaltet und hergestellt, was zu einem weiterlaufenden Kreislauf von unerwünschten, nicht zurücklaufenden Ingenieurkosten bzw. Erstellungskosten führt. Die Verwendung von Rechen-Systemelementen bzw. Rechen-Resourcen, die auf bestimmte Problemtypen ausgerichtet sind, führt deshalb zu einer ineffizienten Verwendung verfügbaren System-Siliziums, wenn man die sich ändernden Rechenerfordernisse bzw. Rechenbedürfnisse in Betracht zieht. Somit ist es wegen der oben beschriebenen Gründe nicht wünschenswert, wenn man versucht, die Rechenleistung zu erhöhen, indem man spezialisierte Hardware verwendet.

Im Stand der Technik wurden verschiedene Versuche bzw. Anläufe unternommen, um sowohl die Rechenleistung zu erhöhen als auch die problemtypische Anwendbarkeit zu maximieren, indem reprogrammierbare oder rekonfigurierbare Hardware verwendet wird. Ein erster derartiger Anlauf bzw. Versuch gemäß dem Stand der Technik liegt in herunterladbaren Mikrocode-Computerarchitekturen. In einer herunterladbaren Mikrocode-Architektur kann das Verhalten von fixen, nicht rekonfigurierbaren Hardware-Systemelementen selektiv geändert werden, indem eine bestimmte Version eines Mikrocodes verwendet wird.

Ein Beispiel für eine derartige Architektur ist diejenige des IBM-Systems/360. Da die fundamentale bzw. grundlegende Rechen-Hardware in derartigen Systemen nach dem Stand der Technik nicht selbst rekonfigurierbar ist, liefern derartige Systeme keine optimierte Rechenleistungsfähigkeit, wenn man einen breiten Bereich von Problemtypen betrachtet.

Ein zweiter Anlauf bzw. Versuch nach dem Stand der Technik im Hinblick auf sowohl einer Erhöhung der Rechenleistungsfähigkeit als auch einer Maximierung der problemtypischen Anwendbarkeit liegt in der Verwendung einer rekonfigurierbaren Hardware, die mit einem nicht-rekonfigurierbaren Host-Prozessor oder Most-System verbunden ist bzw. damit gekoppelt ist. Diesen Versuch bzw. Anlauf kann man als "zugeordnete rekonfigurierbare Prozessor" – bzw. "Attached Reconfigurable Processor" (ARP) -Architektur kategorisieren, bei welcher ein gewisser Anteil der Hardware innerhalb einer Prozessorguppe, die einem Most zugeordnet ist, rekonfigurierbar ist. Beispiele gegenwärtiger ARP-Systeme, die eine Gruppe rekonfigurierbarer Prozessoren verwenden, die mit einem Most-System gekoppelt sind bzw. verbunden sind, beinhalten: die SPLASH-1- und SPLASH-2-Systeme, die an dem Supercomputer Forschungszentrum bzw. Supercomputing Research Center (Bowie, MD, USA) designed bzw. gestaltet wurden; den WILDFIRE bzw. allgemein konfigurierbaren Computer, der von den Annapolis Micro Systems (Annapolis, MD, USA) hergestellt wird, der eine kommerzielle Version der SPLASH-2 darstellt; und dem EVC-1, der von der Virtual Computer Corporation (Reseda, CA, USA), hergestellt wird. Bei den meisten rechenintensiven Problemen wird eine beträchtliche Menge an Zeit darauf verwendet, relativ kleine Abschnitte von Programmcodes auszuführen. Im allgemeinen werden ARP-Architekturen verwendet, um eine rekonfigurierbare Rechenbeschleunigungseinrichtung für derartige Abschnitte von Programmcodes bereitzustellen. Unglücklicherweise leidet ein Rechenmodell, das auf einer oder mehreren rekonfigurierbaren Rechenbeschleunigungseinrichtungen basiert, an beträchtlichen Nachteilen, wie im folgenden beschrieben werden wird.

Ein erster Nachteil der ARP-Architekturen tritt auf, weil ARP-Systeme versuchen, eine optimierte Implementation eines bestimmten Algorithmus in einer rekonfigurierbaren Hardware zu einer bestimmten Zeit bereitzustellen. Die Philosophie, die hinter dem EVC-1 der Virtual Computer Corporation liegt, besteht z. B. darin, einen speziellen Algorithmus in eine spezielle Konfiguration von rekonfigurierbaren Hardware-Systemelementen umzuwandeln, um eine optimierte Rechenleistung für jenen bestimmten Algorithmus bereitzustellen. 5

Rekonfigurierbare Hardware-Systemelemente werden allein zu dem Zweck verwendet, eine optimale Leistungsfähigkeit für einen bestimmten Algorithmus bereitzustellen. Die Verwendung von rekonfigurierbaren Hardware-Systemelementen für allgemeinere Zwecke, wie z. B. dem Management der Ausführung von Instruktionen bzw. Befehlen, wird vermieden. Für einen gegebenen Algorithmus werden somit rekonfigurierbare Hardware-Systemelemente unter dem Gesichtspunkt von individuellen Gattern bzw. Gates betrachtet, die gekoppelt sind, um eine optimale Leistungsfähigkeit zu gewährleisten. 10

Gewisse ARP-Systeme verlassen sich auf ein Programmierungsmodell, bei welchem ein "Programm" sowohl konventionelle Programminstruktionen bzw. -befehle beinhaltet als auch Instruktionen für spezielle Zwecke, die spezifizieren, wie verschiedene rekonfigurierbare Hardware-Systemelemente untereinander verbunden sind. Weil ARP-Systeme rekonfigurierbare Hardware-Systemelemente in einer algorithmus-spezifischen Art und Weise auf Gatterebene betrachten, müssen diese auf einen speziellen Zweck gerichtete Instruktionen explizite Details bezüglich der Natur eines jeden verwendeten rekonfigurierbaren Hardware-Systemelements und der Art und Weise, in der es mit anderen rekonfigurierbaren Hardware-Systemelementen verbunden ist, bereitstellen. Dies beeinflußt nachteilig die Programmkomplexität. Um die Programmkomplexität zu verringern, wurden Versuche unternommen, eine Programmiermodell zu verwenden, bei welchem ein Programm sowohl konventionelle Instruktionen auf hohem Niveau einer Programmiersprache als auch Instruktionen auf hohem Niveau für spezielle Zwecke beinhaltet. Gegenwärtige ARP-Systeme versuchen deshalb, ein Kompilier-System zu verwenden, das in der Lage ist, sowohl Befehle auf hohem Niveau einer Programmiersprache zu kompilieren als auch die zuvor erwähnten Befehle auf hohem Niveau für spezielle Zwecke. Die angestrebte Ausgabe bzw. Sollausgabe eines derartigen Kompilier-Systems ist ein Code in Assemblersprache für konventionelle Instruktionen auf hohem Niveau einer Programmiersprache und in ein Code einer Hardware-Beschreibungssprache bzw. einer "Hardware Description Language" (HDL) für Befehle zu speziellen Zwecken. Unglücklicherweise stellt die automatische Bestimmung einer Gruppe von rekonfigurierbaren Hardware-Systemelementen und ein Schema bezüglich ihrer Verbindung, um eine optimale Rechenleistung für jeglichen bestimmten Algorithmus, der gerade betrachtet wird, ein NP-Hard-Problem bzw. "NP-hard"-Problem dar. Ein Ziel gewisser ARP-Systeme ist die Entwicklung eines Kompilierersystems, das einen Algorithmus direkt in ein optimiertes Schema bezüglich der Verbindungen untereinander für eine Gruppe von Gattern kompilieren kann. Die Entwicklung eines derartigen Kompilierersystems ist jedoch eine außerordentlich schwierige Aufgabe, insbesondere wenn man die vielfachen Typen von Algorithmen betrachtet. 15 20 25 30 35

Ein zweiter Nachteil von ARP-Architekturen tritt auf, weil ein ARP-Apparat die Berechnungsarbeit, die mit dem Algorithmus verbunden ist, für den er konfiguriert worden ist, über eine Vielzahl rekonfigurierbarer Logikvorrichtungen verteilt. Zum Beispiel wird bei einem ARP-Apparat, der realisiert wurde, indem eine Gruppe von feldprogrammierbaren Logikvorrichtungen bzw. "Field Programmable Logic Devices" (FPGAs) verwendet wurde, und der konfiguriert wurde, um eine parallele Multiplikations-Beschleunigungseinrichtung zu implementieren, die Berechnungsarbeit, die mit der parallelen Multiplikation verbunden ist, über die gesamte Gruppe der FPGAs verteilt. Deshalb ist die Größe des Algorithmus, bezüglich dem der ARP-Apparat konfiguriert werden kann, auf die Anzahl der vorliegenden rekonfigurierbaren Logikvorrichtungen beschränkt. In ähnlicher Weise ist die maximale Datensatzgröße, die der ARP-Apparat handhaben kann, beschränkt. Eine Untersuchung von Sourcecodes bzw. Quellcodes lieferten nicht notwendigerweise einen klaren Hinweis auf die Beschränkungen des ARP-Apparats, weil einige Algorithmen Daten-Abhängigkeiten aufweisen können. Im allgemeinen werden datenabhängige Algorithmen vermieden. 40 45

Da weiter ARP-Architekturen die Verteilung von Rechenarbeit über mehrere rekonfigurierbare Logikvorrichtung lehren, erfordert die Anpassung eines neuen (oder sogar leicht modifizierten) Algorithmus, daß die Rekonfiguration massiv durchgeführt werden muß, d. h. die mehreren rekonfigurierbaren Logikvorrichtungen müssen rekonfiguriert werden. Dies beschränkt die maximale Rate, bei der eine Rekonfiguration für alternative Probleme oder Kaskaden-Unterprobleme bzw. hintereinandergeschaltete Unterprobleme auftreten kann. 50

Ein dritter Nachteil von ARP-Architekturen ergibt sich aus der Tatsache, daß eine oder mehrere Abschnitte des Programmcodes auf dem Host ausgeführt werden. Das heißt ein ARP-Apparat ist nicht selbst ein unabhängiges Computersystem bzw. Rechensystem, der ARP-Apparat führt nicht ganze Programme aus und es ist deshalb eine Wechselwirkung mit dem Host notwendig. Da etwas von dem Programmcode auf dem nicht rekonfigurierbaren Host ausgeführt wird, wird die Gruppe bzw. der Satz an verfügbaren Silizium-Resourcen nicht maximal über den Zeitrahmen der Programmausführung ausgenutzt. Insbesondere während der Ausübung von Instruktionen bzw. Befehlen auf Host-Basis, werden die Silizium-Resourcen bzw. Silizium-Systemelemente auf dem ARP-Apparat untätig sein oder ineffizient verwendet werden. In ähnlicher Weise werden, wenn der ARP-Apparat mit Daten arbeitet, die Silizium-Resourcen bzw. die Silizium-Systemelemente auf dem Host im allgemeinen ineffizient verwendet werden. Um leicht mehrere ganze Programme auszuführen, müssen die Silizium-Resourcen bzw. die Silizium-Systemeinheiten in leicht wiederverwendbare Resourcen bzw. Systemeinheiten kopiert werden. Wie vorstehend beschrieben wurde, behandeln ARP-System rekonfigurierbare Hardware-Systemelemente als eine Gruppe von Gattern bzw. Gates, die optimal untereinander verbunden sind, um einen bestimmten Algorithmus zu einer bestimmten Zeit zu implementieren. Somit liefern ARP-Systeme nicht eine Einrichtung, um einen bestimmten Satz bzw. eine bestimmte Gruppe an rekonfigurierbaren Hardware-Systemelementen als ein leicht von einem Algorithmus zum anderen wiederverwendbares Systemelement zu behandeln, weil die Wiederverwendbarkeit ein gewisses Niveau an Unabhängigkeit bezüglich des Algorithmus erfordert. 55 60 65

Ein ARP-Apparat kann nicht das gegenwärtig ausgeführte Hostprogramm als Daten behandeln und kann im allgemeinen nicht sich selbst kontextualisieren. Ein ARP-Apparat kann nicht leicht so hergestellt werden, daß er sich selbst durch das Ausführen seines eigenen Hostprogramms simuliert. Weiter kann ein ARP-Apparat nicht hergestellt werden, um seine eigene HDL oder seine eigenen Anwendungsprogramme auf ihn selbst zu kompilieren, wobei er direkt die rekonfigurierbaren Hardware-Systemelemente bzw. Hardware-Ressourcen verwendet, aus denen er aufgebaut ist. Ein ARP-Apparat ist somit bezüglich seiner Architektur bezüglich in sich geschlossener Rechenmodelle beschränkt, die die Unabhängigkeit von einem Hostprozessor lehren.

Weil ein ARP-Apparat als eine Rechen-Beschleunigungseinrichtung wirkt, ist er im allgemeinen nicht in der Lage, eine unabhängige Eingangs-/Ausgangs- bzw. "Input/Output" (I/O)-Verarbeitung durchzuführen. Typischerweise erfordert ein ARP-Apparat eine Wechselwirkung mit dem Host für eine I/O-Verarbeitung. Die Leistungsfähigkeit eines ARP-Apparats kann deshalb bezüglich des I/O beschränkt sein. Fachleute werden erkennen, daß ein ARP-Apparat jedoch konfiguriert werden kann, um ein spezielles I/O-Problem zu beschleunigen. Da jedoch der gesamte ARP-Apparat auf ein einziges, spezielles Problem hin gestaltet bzw. konfiguriert ist, kann ein ARP-Apparat die I/O-Verarbeitung nicht mit der Datenverarbeitung balancieren, ohne bezüglich des einen oder des anderen einen Kompromiß einzugehen. Darüber hinaus stellt ein ARP-Apparat keine Einrichtung für die Interruptverarbeitung bereit. Die Lehren bezüglich eines ARP's bieten keine derartigen Mechanismen an, da sie auf eine maximale Beschleunigung des Rechnens hin ausgerichtet sind und die Unterbrechung sich negativ auf die Rechenbeschleunigung auswirkt.

Ein vierter Nachteil von ARP-Architekturen existiert, da es Software-Applikationen gibt, die eine inhärente Datenparallelität besitzen, wobei es schwierig ist, diese auszunutzen, indem ein ARP-Apparat verwendet wird. HDL-Komplizierungs-Anwendungen liefern ein derartiges Beispiel, wenn eine Netz-Namen-Symbolauflösung in einer sehr großen Netzliste benötigt wird.

Ein fünfter Nachteil, der mit ARP-Architekturen verbunden ist, ist, daß es im wesentlichen ein SIMD-Computerarchitekturmodell gibt. ARP-Architekturen sind deshalb weniger bezüglich ihrer Architektur effektiv, als ein oder mehrere innovative nicht-rekonfigurierbare Systeme nach dem Stand der Technik. ARP-Systeme spiegeln nur einen Teil des Prozesses der Ausführung eines Programms wider, hauptsächlich die arithmetische Logik für eine arithmetische Berechnung, und zwar für jeden spezifischen Konfigurationsfall, und zwar für so viel Rechenleistung, wie die verfügbare rekonfigurierbare Hardware liefern kann. Im Gegensatz dazu nutzte nach dem Systemdesign der SYMBOL-Maschine bei Fairchild 1971 der gesamte Computer einen einzigen Hardware-Kontext für jeden Aspekt der Programmausführung. Infolgedessen umfaßte SYMBOL jedes Element für die Systemanwendung eines Computers, einschließlich des Hostabschnittes, der durch ARP-Systeme gelehrt bzw. angewiesen wird.

ARP-Architekturen weisen andere Unzulänglichkeiten ebenso auf. Zum Beispiel fehlt es einem ARP-Apparat an einer effektiven Einrichtung, um ein unabhängiges Timing bzw. eine unabhängige Zeitsteuerung für vielfach rekonfigurierbare Logikvorrichtungen bereitzustellen. Ähnlich fehlt es einem kaskadierten ARP-Apparat an einer wirksamen Takt-Verteilungseinrichtung, um unabhängig getaktete Einheiten bzw. Einheiten, die bezüglich ihrer Zeitsteuerung unabhängig sind, bereitzustellen. Nach einem anderen Beispiel ist es schwierig, genau die Ausführungszeit mit den Quellencode-Anweisungen zu korrelieren, für die eine Beschleunigung beabsichtigt wird. Für eine genaue Abschätzung der Netzsystem-Taktrate muß die ARP-Vorrichtung mit einem Werkzeug für computerunterstütztes Design bzw. mit einem "Computer-Aided Design (CAD)"-Tool nach einer HDL-Komplizierung modelliert werden, dies ist ein zeitaufbrauchender Prozeß, um zu einem derartigen Basisparameter zu gelangen.

Was gebraucht wird, ist eine Einrichtung zum rekonfigurierbaren Rechnen, die die Beschränkungen des oben beschriebenen Standes der Technik überwindet.

Die vorliegende Erfindung betrifft ein System und ein Verfahren für skalierbares, paralleles, dynamisch rekonfigurierbares Rechnen. Das System weist wenigstens eine S-Maschine, eine T-Maschine, die jeder S-Maschine entspricht, eine Allzweck-Verbindungsmautrix (GPIM), eine Gruppe von I/O-T-Maschinen, eine oder mehrere I/O-Vorrichtungen und eine Master-Zeitbasis-Einheit auf. In der bevorzugten Ausführungsform beinhaltet das System mehrere S-Maschinen. Jede S-Maschine weist einen Eingang bzw. einen Ausgang auf, der mit einem Ausgang bzw. einem Eingang der entsprechenden T-Maschine verbunden ist. Jede T-Maschine beinhaltet einen Leitweg-Eingang bzw. Führungs-Eingang und einen Leitweg-Ausgang bzw. Führungs-Ausgang, der mit der GPIM verbunden ist, wie dies auch bei jeder I/O-T-Maschine der Fall ist. Eine I/O-T-Maschine beinhaltet weiter einen Eingang und einen Ausgang, der mit einer I/O-Vorrichtung verbunden ist. Schließlich weist jede S-Maschine, T-Maschine und I/O-T-Maschine einen Master-Zeitsteuer-Eingang bzw. einen Master-Timing-Eingang auf, der mit einem Zeitsteuer-Ausgang bzw. Timing-Ausgang der Master-Zeitbasisseinheit verbunden ist.

Die Master-Zeitbasisseinheit liefert eine systemweite Frequenzreferenz an jede S-Maschine, T-Maschine und I/O-T-Maschine. Jede S-Maschine ist ein Computer, der eine Verarbeitungseinheit aufweist, die selektiv rekonfiguriert werden kann, und zwar während der Ausführung von Programmstrukturen. Jede T-Maschine ist eine Datentransfervorrichtung bzw. Datenübertragungsvorrichtung. Die GPIM liefert eine skalierbare Punkt-zu-Punkt-Parallel-Verbindungseinrichtung zur Kommunikation zwischen T-Maschinen. Zusammengenommen stellt der Satz an T-Maschine und die GPIM eine skalierbare Punkt-zu-Punkt-Parallelverbindungseinrichtung zur Kommunikation zwischen S-Maschinen bereit.

Eine S-Maschine umfaßt vorzugsweise eine erste lokale Zeitbasisseinheit, einen Speicher und eine dynamisch rekonfigurierbare Verarbeitungseinheit bzw. eine "Dynamically Reconfigurable Processing Unit" (DRPU). Die erste lokale Zeitbasisseinheit umfaßt einen Zeitsteuerungseingang bzw. einen Timingeingang, der mit der Master-Zeitbasisseinheit verbunden ist, und einen Zeitsteuerungsausgang bzw. Timingausgang, der mit einem Zeitsteuerungseingang der DRPU und einem Zeitsteuerungseingang des Speichers über die erste Zeitsteuerungs-Signalleitung verbunden ist. Die DRPU umfaßt einen Steuersignalausgang, einen Adressenausgang bzw. einen bidirek-

tionalen Datenport, der mit einem Steuersignaleingang, einem Adresseneingang bzw. einem bidirektionalen Datenport des Speichers über eine Speichersteuerleitung, eine Adressenleitung bzw. eine Speicher-I/O-Leitung verbunden ist. Die DRPU weist ebenso einen bidirektionalen Steuerport auf, der mit einem bidirektionalen Steuerport ihrer entsprechenden T-Maschine über eine externe Steuerleitung verbunden ist.

Die erste lokale Zeitbasiseinheit empfängt ein Master-Zeitsteuersignal von der Master-Zeitbasiseinheit und erzeugt ein erstes lokales Zeitsteuersignal, das zu der DRPU und dem Speicher über eine erste Zeitsteuersignalleitung geliefert wird. Bei dem Speicher handelt es sich vorzugsweise um einen Speicher mit wahlfreiem Zugriff bzw. ein "Random Access Memory" (RAM), das Programminstruktionen, Programmdaten und eine oder mehrere Konfigurationsdatensätze speichert. Bei der bevorzugten Ausführungsform ist ein gegebener S-Maschinen-
5 speicher für jede andere S-Maschine in dem System über die GPIM und ihre korrespondierende T-Maschine zugänglich.

10 Eine Gruppe von Programminstruktionen bzw. Programmbefehlen, die darauf ausgerichtet sind bzw. spezialisiert sind, einen spezifischen Satz von Befehlen bezüglich potentiell großen Datensätze durchzuführen, wird hierin als ein "Innenschleifen"- bzw. "Inner-Loop"-Abschnitt eines Programms bezeichnet. Eine Gruppe von Programminstruktionen, die für das Durchführen von Allzweckoperationen und/oder für das Übertragen der Steuerung von einem Innenschleifenabschnitt zu einem anderen verantwortlich ist, wird hierin als "Außenschleifen"- bzw. "Outer-Loop"-Abschnitt des Programms bezeichnet. Innerhalb jedes gegebenen Programms besteht 15 jeder Innenschleifenabschnitt vorzugsweise aus einer kleinen Anzahl von Instruktionstypen, während Außenschleifenabschnitte vorzugsweise eine Mannigfaltigkeit bzw. Vielzahl von Allzweck-Instruktionstypen beinhalten.

20 Jeder Konfigurationsdatensatz, der in dem Speicher gespeichert ist, spezifiziert eine DRPU-Hardware-Organisation, die für die Implementation bzw. Realisierung einer entsprechenden Befehlssatzarchitektur bzw. Instruktionssatzarchitektur ("Instruction Set Architecture" bzw. ISA) optimiert ist. Eine ISA ist ein primitiver Satz von Instruktionen, der verwendet werden kann, um einen Computer zu programmieren. Gemäß der vorliegenden Erfindung kann eine ISA kategorisiert werden als eine Innenschleifen-ISA oder eine Außenschleifen-ISA, und zwar gemäß der Anzahl und Typen von Instruktionen, die sie enthält. Eine Innenschleifen-ISA besteht aus relativ wenig Instruktionen, und zwar wo die Instruktionen nützlich sind, um spezifische Typen von Operationen durchzuführen. Eine Außenschleifen-ISA beinhaltet mehrere Instruktionen, und zwar wo die Instruktionen nützlich sind, um eine Vielzahl von Allzweck-Operationen durchzuführen.

25 Programminstruktionen, die in dem Speicher gespeichert sind, beinhalten selektiv eine oder mehrere Rekonfigurations-Anweisungen, und zwar wo jede Rekonfigurationsanweisung auf einen Konfigurationsdatensatz Bezug nimmt. Während der Programmausführung durch die DRPU können eine oder mehrere Rekonfigurationsanweisungen gewählt werden. Die Auswahl von gegebenen Rekonfigurationsanweisungen führt zu einer Rekonfiguration der DRPU-Hardware gemäß dem Konfigurationsdatensatz, auf dem durch die Rekonfigurationsanweisung Bezug genommen wird. Somit wird nach der Auswahl einer Rekonfigurationsanweisung die DRPU-Hardware rekonfiguriert, um eine optimierte Implementation einer bestimmten ISA bereitzustellen. Gemäß der vorliegenden Erfindung wird eine Rekonfiguration der DRPU ebenso in Antwort auf einen Rekonfigurationsinterrupt ausgelöst, und zwar wo der Rekonfigurationsinterrupt auf einen Konfigurationsdatensatz Bezug nimmt, der einer ISA in der oben beschriebenen Art und Weise entspricht.

30 Die DRPU umfaßt eine Instruktionsabrufeinheit bzw. "Instruction Fetch Unit" (IFU), eine Datenoperationseinheit bzw. "Data Operate Unit" (DOU) und eine Adressenoperationseinheit bzw. "Address Operate Unit" (AOU), wobei jede dynamisch rekonfigurierbar ist. Bei der bevorzugten Ausführungsform wird die DRPU implementiert, indem eine rekonfigurierbare Logikvorrichtung, wie z. B. ein Xilinx XC4013 programmierbares Gatter-Array bzw. "Programmable Gate Array" (FPGA) verwendet wird. Die reprogrammierbare Logikvorrichtung liefert vorzugsweise eine Vielzahl von selektiv reprogrammierbaren 1) Logikblöcken oder konfigurierbaren Logikblöcken bzw. "Configurable Logic Blocks" (CLBs); 2) I/O-Blöcke (IOBs); 3) Verbindungsstrukturen; 4) Datenspeicher-Systemeinheiten; 5) Dreizustands-Puffer-Systemeinheiten; und 6) Fähigkeiten einer fest verdrahteten Logik.

40 Die IFU umfaßt einen Speichersteuerausgang, der den Speichersteuerausgang der DRPU bildet, einen Dateneingang, der mit der Speicher-I/O-Leitung verbunden ist und einen bidirektionalen Steuerport, der den bidirektionalen Steuerport der DRPU bildet. Die IFU umfaßt zusätzlich einen ersten, zweiten und dritten Steuerausgang. Die DOU und die AOU umfassen jeweils einen bidirektionalen Datenport, der mit der Speicher-I/O-Leitung verbunden ist, und die AOU umfaßt einen Adressenausgang, der mit der Adressenleitung verbunden ist. Die DOU umfaßt einen ersten Steuereingang, der mit dem ersten Steuerausgang der IFU über eine erste Steuerleitung verbunden ist. Die AOU umfaßt einen ersten Steuereingang, der mit dem zweiten Steuerausgang der IFU über eine zweite Steuerleitung verbunden ist. Sowohl die DOU als auch die AOU umfassen einen zweiten Steuereingang, der mit dem dritten Steuerausgang der IFU über eine dritte Steuerleitung verbunden ist. Schließlich umfassen sowohl die IFU als auch DOU und die AOU einen Zeitsteuereingang, der mit der ersten Zeitsteuersignalleitung verbunden ist.

45 Die IFU verwaltet Instruktionenabruf- und Instruktionen-Decodieroperationen, Speicherzugriffsoperationen, DRPU-Rekonfigurationsoperationen und gibt Steuersignale zu der DOU und der AOU aus, um die Ausführungen der Instruktionen zu erleichtern. Die IFU umfaßt vorzugsweise einen Architekturbeschreibungsspeicher, eine Instruktionszustands-Folgesteuerungseinheit bzw. einen "Instruction State Sequencer" (ISS), eine Speicherzugriffslogik, eine Rekonfigurationslogik, eine Interruptlogik, eine Abrufsteuereinheit, einen Instruktionspuffer, eine Decodiersteuereinheit, einen Instruktionendecoder, einen Opcode- bzw. Operationscode-Speicherregister-
55 satz, einen Registerdatei- bzw. Registerfile-(RF)-Adressenregistersatz, einen Konstantenregistersatz und einen Prozeßsteuerregistersatz. Die ISS umfaßt einen ersten bzw. einen zweiten Steuerausgang, der die ersten bzw. zweiten Steuerausgänge der IFU umfaßt; einen Zeitsteuerungseingang, der den Zeitsteuerungseingang der IFU

ausbildet; einen Abruf-/Decodiersteuerausgang, der mit einem Steuereingang der Abrufsteuereinheit und einem Steuereingang des Decodiersteuereingangs verbunden ist; einen bidirektionalen Steuerport, der mit einem ersten bidirektionalen Steuerport sowohl der Speicherzugriffslogik als auch der Rekonfigurationslogik und der Interruptlogik verbunden ist; einen Operationscodeeingang, der mit einem Ausgang des Operationscodespeicherregistersatzes verbunden ist; einen bidirektionalen Datenport, der mit einem bidirektionalen Datenport des Prozeßsteuerregistersatzes verbunden ist. Sowohl die Speicherzugriffslogik als auch die Rekonfigurationslogik und die Interruptlogik umfaßt einen zweiten bidirektionalen Steuerport, der mit der externen Steuerleitung verbunden ist; einen Dateneingang, der mit einem Datenausgang des Architekturbeschreibungsspeichers verbunden ist. Die Speicherzugriffslogik umfaßt ebenso einen Steuerausgang, der den Speichersteuerausgang des IFU ausbildet und die Interruptlogik umfaßt zusätzlich einen Ausgang, der mit dem bidirektionalen Datenport des Verarbeitungs- bzw. Prozeßsteuerregistersatzes verbunden ist.

Der Architektur-Beschreibungsspeicher umfaßt vorzugsweise einen Speicher zum Speichern der Architektur-Spezifikationssignale, die die DRPU-Konfiguration zu jeder gegebenen Zeit kennzeichnen. Die Architektur-Spezifikationssignale beinhalten vorzugsweise eine Referenz zu einem vorgegebenen Konfigurationsdatensatz bzw. Default-Datensatz; eine Referenz bzw. einen Bezug auf eine Liste von ermöglichen Konfigurationsdatensätzen; ein atomares Speicher-Adresseninkrement; und einen Satz von Interrupt-Antwortsignalen, die spezifizieren, wie die gegenwärtige DRPU-Hardware-Konfiguration auf die Interrupts antwortet. Die ISS umfaßt vorzugsweise eine Zustandsmaschine, die das Ausführen von Instruktionen innerhalb der gegenwärtig betrachteten ISA erleichtert, indem Signale zu der Abrufsteuereinheit, der Decodiersteuereinheit, der DOU, der AOU und der Speicherzugriffslogik ausgegeben werden. Die ISS gibt DOU-Steuersignale auf die erste Steuerleitung, AOU-Steuersignale auf die zweite Steuerleitung und RF-Adressen und -Konstanten auf die dritte Steuerleitung. Die Interruptlogik umfaßt vorzugsweise eine Zustandsmaschine, die vorzugsweise Interrupt-Meldungsoperationen durchführt. Die Rekonfigurationslogik umfaßt vorzugsweise eine Zustandsmaschine, die Rekonfigurations-Operationen in Antwort auf ein Rekonfigurationssignal durchführt. In der bevorzugten Ausführungsform wird das Rekonfigurationssignal in Antwort auf einen Rekonfigurationsinterrupt oder, wenn eine Rekonfigurationsanweisung ausgewählt wird, während der Programmausführung erzeugt.

Die DOU führt Operationen aus, die zu Datenberechnungen in Bezug stehen, und zwar in Übereinstimmung mit DOU-Steuersignalen, RF-Adressen und Konstanten, die von der IFU empfangen werden. Die DOU umfaßt vorzugsweise einen DOU-Crossbar-Schalter, eine Abspeicher-/Ausrichtlogik und eine Datenoperationslogik. Der DOU-Crossbar-Schalter ("X-Bar-Schalter") umfaßt einen bidirektionalen Datenport, der den bidirektionalen Datenport der DOU ausbildet; einen Konstanteneingang, der mit der dritten Steuerleitung der IFU verbunden ist; einen ersten Daten-Feedback-Eingang bzw. Daten-Rückführ-Eingang, der mit einem Datenausgang der Datenoperationslogik verbunden ist; einen zweiten Daten-Feedback-Eingang bzw. Daten-Rückführ-Eingang, der mit einem Datenausgang der Abspeicher-/Ausrichtlogik verbunden ist; einen Datenausgang, der mit einem Dateneingang der Abspeicher-/Ausrichtlogik verbunden ist. Die Abspeicher-/Ausrichtlogik beinhaltet einen Adresseneingang, der mit der dritten Steuerleitung verbunden ist, und die Datenoperationslogik beinhaltet einen Dateneingang, der mit dem Ausgang der Abspeicher-/Ausrichtlogik verbunden ist. Schließlich umfaßt sowohl der DOU-Crossbar-Schalter als auch die Abspeicher-/Ausrichtlogik und die Datenoperationslogik einen Steuer-eingang, der mit der ersten Steuerleitung verbunden ist.

Der DOU-Crossbar-Schalter lädt die Daten von dem Speicher, überträgt Ergebnisse, die von der Datenoperationslogik ausgegeben werden, zu der Abspeicher-/Ausrichtlogik oder dem Speicher und lädt Konstante, die durch die IFU in Antwort auf die DOU-Steuersignale, die an ihrem Eingang empfangen werden, ausgegeben werden. Die Abspeicher-/Ausrichtlogik stellt eine temporäre Speicherung für Operanden, Konstante und Teilergebnisse bereit, die mit den Datenberechnungen verbunden sind. Die Datenoperationslogik führt arithmetische, Schiebe- und/oder logische Operationen in Antwort auf die DOU-Steuersignale, die bei ihrem Steuereingang empfangen werden, durch.

Die AOU führt Operationen durch, die in Beziehung zu der Adressberechnung stehen, und umfaßt vorzugsweise einen AOU-Crossbar-Schalter, eine Abspeicher-/Zähllogik, eine Adressenoperationslogik und einen Adressenmultiplexer. Der AOU-Crossbarschalter umfaßt einen bidirektionalen Datenport, der den bidirektionalen Datenport der AOU ausbildet; einen Adressen-Rückführ-Eingang, der mit einem Adressenausgang der Adressenoperationslogik verbunden ist; einen Konstanteneingang, der mit der dritten Steuerleitung verbunden ist; und einen Adressenausgang, der mit einem Adresseneingang und der Abspeicher-/Zähllogik verbunden ist. Die Abspeicher-/Zähllogik beinhaltet einen RF-Adresseneingang, der mit der dritten Steuerleitung verbunden ist, und einen Adressenausgang, der mit einem Adresseneingang der Adressenoperationslogik verbunden ist. Der Adressenmultiplexer umfaßt einen ersten Eingang, der mit dem Adressenausgang der Abspeicher-/Zähllogik verbunden ist, und einen zweiten Eingang, der mit dem Adressenausgang der Adressenoperationslogik verbunden ist. Sowohl der AOU-Crossbar-Schalter als auch die Abspeicher-/Zähllogik und die Adressenoperationslogik umfassen einen Steuereingang, der mit der zweiten Steuerleitung verbunden ist.

Der AOU-Steuerschalter lädt Adressen aus dem Speicher, überträgt Ergebnisse, die von der Adressenoperationslogik ausgegeben wurden, zu der Abspeicher-/Zähllogik oder dem Speicher und lädt Konstante, die von der IFU ausgegeben wurden, in die Abspeicher-/Zähllogik in Antwort auf die AOU-Steuersignale, die an seinem Eingang empfangen werden. Die Abspeicher-/Zähllogik stellt eine temporäre Speicherung von Adressen und Adressenberechnungsergebnissen bereit. Die Adressenoperationslogik führt arithmetische Operationen bezüglich der Adressen in Übereinstimmung mit den AOU-Steuersignalen durch, die bei ihrem Steuereingang empfangen werden. Der Adressenmultiplexer wählt selektiv eine Adresse aus, die von der Abspeicher-/Zähllogik oder der Adressenoperationslogik empfangen werden, aus, und zwar in Übereinstimmung mit den AOU-Steuersignalen, die bei seinem Steuereingang empfangen werden.

Jedes Element innerhalb der IFU, der DOU und der AOU wird implementiert bzw. realisiert, indem rekonfigu-

rierbare Hardware-Systemelemente innerhalb der rekonfigurierbaren Logikvorrichtung verwendet werden, wie es durch einen gegebenen Konfigurationsdatensatz spezifiziert ist, der einer bestimmten ISA entspricht. Die detaillierte interne Struktur der Elemente innerhalb der IFU, der DOU und der AOU variiert vorzugsweise in Abhängigkeit von dem Typ der ISA, für den die DRPU konfiguriert ist, um zu jedem gegebenen Augenblick zu implementieren bzw. zu realisieren. Für eine Außenschleifen-ISA ist die interne Struktur eines jeden Elements innerhalb der IFU, der DOU und der AOU vorzugsweise für die serielle Instruktionsverarbeitung optimiert. Für eine Innenschleifen-ISA ist die interne Struktur eines jeden Elements innerhalb der IFU, der DOU und der AOU vorzugsweise für die parallele Instruktionsverarbeitung optimiert.

Jede T-Maschine umfaßt vorzugsweise eine gemeinsame Schnittstellen- und Steuereinheit bzw. "common interface control unit", einen Satz an Verbindungs-I/O-Einheiten und eine zweite lokale Zeitbasiseinheit. Die zweite lokale Zeitbasiseinheit umfaßt einen Zeitsteuerungseingang, der mit der Master-Zeitbasiseinheit verbunden ist, und einen Zeitsteuerungsausgang, der mit einem Zeitsteuerungseingang der gemeinsamen Schnittstellen- und Steuereinheit verbunden ist. Die gemeinsame Schnittstellen- und Steuereinheit umfaßt einen Adressenausgang, der mit der Adressenleitung verbunden ist, einen ersten bidirektionalen Datenport, der mit der Speicher-I/O-Leitung verbunden ist, einen bidirektionalen Steuerport, der mit der externen Steuerleitung verbunden ist, und einem zweiten bidirektionalen Datenport, der mit einem bidirektionalen Datenport eines jeden seiner assoziierten Verbindungs-I/O-Einheiten verbunden ist.

Die zweite lokale Zeitbasiseinheit erzeugt ein zweites lokales Zeitbasissignal, das von der Master-Frequenzreferenz abgeleitet wird, die von der Master-Zeitbasiseinheit empfangen wird. Die gemeinsame Schnittstellen- und Steuereinheit verwaltet den Transfer von Daten und Befehlen zwischen seiner entsprechenden S-Maschine und einer ihrer zugeordneten Verbindungs-I/O-Einheiten. Jede Verbindungs-I/O-Einheit überträgt Nachrichten bzw. Meldungen, die von ihrer zugeordneten gemeinsamen Schnittstellen- und Steuereinheit empfangen werden, zu einer anderen Verbindungs-I/O-Einheit über die GPIM. Jede Verbindungs-I/O-Einheit überträgt ebenso selektiv Nachrichten, die von anderen Verbindungs-I/O-Einheiten empfangen werden, zu ihrer zugeordneten gemeinsamen Schnittstellen- und Steuereinheit.

Jede I/O-T-Maschine umfaßt vorzugsweise eine gemeinsame Kunden-Schnittstellen- und Steuereinheit, eine Verbindungs-I/O-Einheit und eine dritte lokale Zeitbasiseinheit. Die internen Verbindungen bzw. Kopplungen innerhalb einer I/O-T-Maschine sind zu jenen innerhalb einer T-Maschine analog; jedoch wird eine I/O-T-Maschine eher mit einer I/O-Vorrichtung bzw. ein I/O-Bauelement verbunden als mit einer S-Maschine, sie enthält deshalb Verbindungen, die für eine bestimmte I/O-Vorrichtung spezifisch sind. Über ihre entsprechende T-Maschine, die GPIM und eine I/O-T-Maschine kommuniziert eine S-Maschine mit einer bestimmten I/O-Vorrichtung in dem System.

Die GPIM liefert eine skalierbare Punkt-zu-Punkt-Verbindungseinrichtung für eine parallele Kommunikation zwischen T-Maschinen. Der Satz an T-Maschinen und die GPIM bilden zusammen eine skalierbare Punkt-zu-Punkt-Verbindungseinrichtung für parallele Kommunikation zwischen S-Maschinen. Die GPIM umfaßt vorzugsweise ein "k-ary, n-cube" bzw. k-n-kubisches statisches Verbindungsnetzwerk mit einer Anzahl erster Kommunikationskanäle und einer Anzahl zweiter Kommunikationskanäle. Jeder erste Kommunikationskanal beinhaltet eine Vielzahl von Knotenverbindungsplätzen, wie dies bei jedem zweiten Verbindungskanal der Fall ist. Jede Verbindungs-I/O-Einheit in dem System ist mit der GPIM derart verbunden, daß ihr Eingang mit einem bestimmten Knotenverbindungsplatz über eine Nachrichteneingangsleitung verbunden ist und ihr Ausgang zu einem anderen Knotenverbindungsplatz über eine Nachrichtenausgangsleitung verbunden ist. Die GPIM ist somit ein skalierbares Netzwerk zum parallelen Leiten bzw. Führen von Daten und Befehlen zwischen mehreren Verbindungs-I/O-Einheiten.

Im folgenden werden kurz die Zeichnungen beschrieben:

Fig. 1 ist ein Blockdiagramm einer bevorzugten Ausführungsform eines Systems zum skalierbaren, parallelen, dynamisch rekonfigurierbaren Rechnen, das in Übereinstimmung mit der vorliegenden Erfindung aufgebaut ist;

Fig. 2 ist ein Blockdiagramm einer bevorzugten Ausführungsform einer S-Maschine gemäß der vorliegenden Erfindung;

Fig. 3A ist ein exemplarisches Programmlisting, das Rekonfigurationsanweisungen enthält;

Fig. 3B ist ein Flußdiagramm bekannter Kompilierungsoperationen, die während der Kompilierung einer Folge von Programminstruktionen ausgeführt werden;

Fig. 3C und 3D sind Flußdiagramme der bevorzugten Kompilierungsoperationen, die von einem Kompilator zum dynamisch rekonfigurierbaren Rechnen durchgeführt werden;

Fig. 4 ist ein Blockdiagramm einer bevorzugten Ausführungsform einer dynamisch rekonfigurierbaren Verarbeitungseinheit bzw. einer "Dynamically Reconfigurable Processing Unit" gemäß der vorliegenden Erfindung;

Fig. 5 ist ein Blockdiagramm einer bevorzugten Ausführungsform einer Instruktionsabrufeinheit bzw. einer "Instruction Fetch Unit" gemäß der vorliegenden Erfindung;

Fig. 6 ist ein Zustandsdiagramm, das einen bevorzugten Satz von Zuständen zeigt, die durch eine Befehlszustandsfolgesteuereinheit bzw. einen "Instruction State Sequencer" gemäß der vorliegenden Erfindung unterstützt werden;

Fig. 7 ist ein Zustandsdiagramm, das einen bevorzugten Satz von Zuständen zeigt, die durch eine Interruptlogik der vorliegenden Erfindung unterstützt werden;

Fig. 8 ist ein Blockdiagramm einer bevorzugten Ausführungsform einer Datenoperationseinheit bzw. einer "Data Operate Unit" der vorliegenden Erfindung;

Fig. 9A ist ein Blockdiagramm einer ersten beispielhaften Ausführungsform der Datenoperationseinheit, die für die Implementation bzw. Realisierung einer Instruktionssatz-Architektur bzw. einer "Instruction Set Architecture" gemäß einer Allzweck-Außenschleife konfiguriert ist;

Fig. 9B ist ein Blockdiagramm für eine zweite beispielhafte Ausführungsform der Datenoperationseinheit, die

für die Implementation einer Innenschleifen-Instruktionssatz-Architektur konfiguriert ist;

Fig. 10 ist ein Blockdiagramm einer bevorzugten Ausführungsform für eine Adressenoperationseinheit bzw. für eine "Address Operate Unit" gemäß der vorliegenden Erfindung;

Fig. 11A ist ein Blockdiagramm einer ersten beispielhaften Ausführungsform für die Adressenoperationseinheit, die für die Implementation bzw. Realisierung einer Instruktionssatz-Architektur gemäß einer Allzweck-Außenschleife konfiguriert ist;

Fig. 11B ist ein Blockdiagramm für eine zweite beispielhafte Ausführungsform der Adressenoperationseinheit, die für die Implementation bzw. Verwirklichung einer Innenschleifen-Instruktionssatz-Architektur konfiguriert ist;

Fig. 12A ist ein Diagramm, das eine beispielhafte Zuordnung von rekonfigurierbaren Hardware-Systemeinheiten zwischen der Instruktionsabrufeinheit, der Datenoperationseinheit und der Adressenoperationseinheit für eine Außenschleifen-Instruktionssatz-Architektur zeigt;

Fig. 12B ist ein Diagramm, das eine exemplarische Zuordnung von rekonfigurierbaren Hardware-Systemeinheiten zwischen der Instruktionsabrufeinheit, der Datenoperationseinheit und der Adressenoperationseinheit für eine Innenschleifen-Instruktionssatz-Architektur zeigt;

Fig. 13 ist ein Blockdiagramm einer bevorzugten Ausführungsform einer T-Maschine der vorliegenden Erfindung;

Fig. 14 ist ein Blockdiagramm einer Verbindungs-I/O-Einheit der vorliegenden Erfindung;

Fig. 15 ist ein Blockdiagramm einer bevorzugten Ausführungsform einer I/O-T-Maschine der vorliegenden Erfindung;

Fig. 16 ist ein Blockdiagramm einer bevorzugten Ausführungsform einer Allzweck-Verbindungsmatrix bzw. einer "General Purpose Interconnect Matrix" der vorliegenden Erfindung; und

Fig. 17A und 17B zeigen ein Flußdiagramm eines bevorzugten Verfahrens zum skalierbaren, parallelen, dynamisch rekonfigurierbaren Rechnen in Übereinstimmung mit der vorliegenden Erfindung.

Im folgenden werden die bevorzugten Ausführungsformen detailliert beschrieben:

Fig. 1 zeigt ein Blockdiagramm einer bevorzugten Ausführungsform eines Systems 10 zum skalierbaren, parallelen, dynamisch rekonfigurierbaren Rechnen, das in Übereinstimmung mit der vorliegenden Erfindung aufgebaut ist. Das System 10 umfaßt vorzugsweise wenigstens eine S-Maschine 12, eine T-Maschine 14, die zu jeder S-Maschine 12 korrespondiert, eine Allzweck-Verbindungsmatrix bzw. "General Purpose Interconnect Matrix" (GPIM) 16, wenigstens eine I/O-T-Maschine 18, eine oder mehrere I/O-Vorrichtungen 20 und eine Master-Zeitbasisseinheit 22. Bei der bevorzugten Ausführungsform umfaßt das System 10 mehrere S-Maschinen 12 und somit mehrere T-Maschinen 14 sowie mehrere I/O-T-Maschinen 18 und mehrere I/O-Vorrichtungen 20.

Jede der S-Maschinen 12, T-Maschinen 14 und I/O-T-Maschinen 18 umfaßt einen Master-Zeitsteuerungseingang, der mit einem Zeitsteuerungsausgang der Master-Zeitbasisseinheit 22 verbunden ist.

Jede S-Maschine 12 umfaßt einen Eingang und einen Ausgang, der mit ihrer entsprechenden bzw. korrespondierenden T-Maschine 14 verbunden ist. Zusätzlich dazu, daß der Eingang und der Ausgang mit ihrer entsprechenden bzw. korrespondierenden S-Maschine 12 verbunden ist, umfaßt jede T-Maschine 14 einen Leitwegeingang bzw. Führungseingang und einen Leitwegausgang bzw. Führungsausgang, der mit der GPIM 16 verbunden ist. In ähnlicher Art und Weise umfaßt jede I/O-T-Maschine 18 einen Eingang und einen Ausgang, der mit einer I/O-Vorrichtung 20 verbunden ist, und einen Leitwegeingang und einen Leitwegausgang, der mit der GPIM 16 verbunden ist.

Wie detaillierter weiter unten beschrieben werden wird, handelt es sich bei jeder S-Maschine 12 um einen dynamisch rekonfigurierbaren Computer bzw. Rechner. Die GPIM 16 bildet eine Punkt-zu-Punkt-Parallel-Verbindungseinrichtung, die die Kommunikation zwischen T-Maschinen 14 erleichtert. Der Satz von T-Maschinen 14 und die GPIM 16 bilden eine Punkt-zu-Punkt-Parallel-Verbindungseinrichtung für einen Datentransfer zwischen S-Maschinen 12. In ähnlicher Weise bilden die GPIM 16, der Satz von T-Maschinen 14 und der Satz von I/O-T-Maschinen 18 eine Punkt-zu-Punkt-Parallel-Verbindungseinrichtung für einen I/O-Transfer bzw. für eine I/O-Übertragung zwischen S-Maschinen 12 und jeder I/O-Vorrichtung 20. Die Master-Zeitbasisseinheit 22 umfaßt einen Oszillator, der ein Master-Zeitsteuersignal für jede S-Maschine 12 und T-Maschine 14 liefert.

Bei einer beispielhaften Ausführungsform ist jede S-Maschine 12 realisiert, indem ein Xilinx XC4013 (Xilinx, Inc, San Jose, CA, USA) feldprogrammierbares Gate-Array bzw. "Field Programmable Gate Array" (FPGA) verwendet wird, das mit 64 Megabyte Speicher mit wahlfreiem Zugriff bzw. "Random Access Memory" (RAM) verbunden ist. Jede T-Maschine 14 wird realisiert, indem ungefähr 50% der rekonfigurierbaren Hardware-Systemelemente in einer Xilinx XC4013 FPGA verwendet werden, dies ist auch bei jeder I/O-T-Maschine 18 der Fall. Die GPIM 14 ist als eine toroidale Verbindungsmaße bzw. Verbindungsvermaschung verwirklicht. Die Master-Zeitbasisseinheit 22 ist ein Taktoszillator, der mit dem Taktverteilungsschaltkreis verbunden ist, um eine systemweite Frequenzreferenz zu liefern, wie dies in der US-Patentanmeldung mit dem Titel "System und Verfahren für phasensynchrones Takt und phasensynchrone Nachrichtenübertragung mit flexibler Frequenz" bzw. "System and Method for Phase-Synchronous, Flexible Frequency Clocking and Messaging" beschrieben ist. Vorzugsweise übertragen die GPIM 14, die T-Maschinen 12 und die I/O-T-Maschinen 18 Information in Übereinstimmung mit dem ANSI/IEEE-Standard 1596-1992, der ein skalierbares kohärentes Interface bzw. ein "Scalable Coherent Interface" (SCI) festlegt.

Bei der bevorzugten Ausführungsform umfaßt das System 10 mehrere S-Maschinen 12, die parallel arbeiten. Die Struktur und die Funktionalität bzw. Wirkungsweise einer jeden einzelnen S-Maschine 12 sind weiter unten detailliert unter Bezugnahme auf die Fig. 2 bis 12B beschrieben. Nimmt man nun Bezug auf die Fig. 2, so ist ein Blockdiagramm einer bevorzugten Ausführungsform einer S-Maschine 12 gezeigt. Die S-Maschine 12 umfaßt eine erste lokale Zeitbasisseinheit 30, eine dynamisch rekonfigurierbare Verarbeitungseinheit bzw. eine "Dynamically Reconfigurable Processing Unit" (DRPU) 32, um Programminstruktionen auszuführen, und einen Speicher

34. Die erste lokale Zeitbasiseinheit 30 umfaßt einen Zeitsteuerungseingang, der den Master-Zeitsteuerungseingang der S-Maschine ausbildet. Die erste lokale Zeitbasiseinheit 30 umfaßt ebenso einen Zeitsteuerungsausgang, der ein erstes lokales Zeitsteuerungssignal oder einen ersten lokalen Zeitsteuerungstakt zu einem Zeitsteuerungseingang der DRPU 32 und einem Zeitsteuerungseingang des Speichers 34 über eine erste Zeitsteuerungssignalleitung 40 liefert. Die DRPU 32 umfaßt einen Steuersignalausgang, der mit einem Steuersignaleingang des Speichers 34 über eine Speichersteuerleitung 42 verbunden ist; einen Adressenausgang, der mit einem Adresseneingang des Speichers 34 über eine Adressenleitung 44 verbunden ist; einen bidirektionalen Datenport, der mit einem bidirektionalen Datenport des Speichers 34 über eine Speicher-I/O-Leitung 46 verbunden ist. Die DRPU 32 umfaßt zusätzlich einen bidirektionalen Steuerport, der mit einem bidirektionalen Steuerport ihrer entsprechenden T-Maschine 14 über eine externe Steuerleitung 48 verbunden ist. Wie in der Fig. 2 gezeigt, überspannt die Speichersteuerungsleitung 42 X Bits, die Adressenleitung 44 M Bits, die Speicher-I/O-Leitung 46(N x k) Bits und die externe Steuerleitung 48 überspannt Y Bits.

Bei der bevorzugten Ausführungsform empfängt die erste lokale Zeitbasiseinheit 30 das Master-Zeitsteuerungssignal von der Master-Zeitbasiseinheit 22. Die erste lokale Zeitbasiseinheit 30 erzeugt das erste lokale Zeitsteuerungssignal aus dem Master-Zeitsteuerungssignal und liefert das erste lokale Zeitsteuerungssignal zu der DRPU 32 und dem Speicher 34. Bei der bevorzugten Ausführungsform kann das erste lokale Zeitsteuerungssignal von einer S-Maschine 12 zu der anderen sich ändern. Somit arbeiten die DRPU 32 und der Speicher 34 innerhalb einer gegebenen S-Maschine 12 bei unabhängigen Taktraten relativ zu der DRPU 32 dem Speicher 34 innerhalb irgendeiner anderen S-Maschine 12. Vorzugsweise ist das erste lokale Zeitsteuerungssignal mit dem Master-Zeitsteuerungssignal phasensynchronisiert. Bei der bevorzugten Ausführungsform ist die erste lokale Zeitbasiseinheit 30 realisiert, indem eine phasensynchronisierte Frequenzumwandlungsschaltung verwendet wird, einschließlich einer phasensynchronisierten Detektionsschaltung, die realisiert wird, indem rekonfigurierbare Hardware-Systemeinheiten verwendet werden. Fachleute werden erkennen, daß bei einer alternativen Ausführungsform die erste lokale Zeitbasiseinheit 30 als ein Abschnitt eines Taktverteilungsbaums realisiert werden kann.

Der Speicher 34 wird vorzugsweise als ein RAM realisiert bzw. implementiert und speichert Programminstruktionen, Programmdaten und Konfigurationsdatensätze für die DRPU 32. Der Speicher 34 einer jeden gegebenen S-Maschine 12 ist vorzugsweise für jede andere S-Maschine 12 in dem System 10 über die GPIM 16 zugänglich. Darüber hinaus ist jede S-Maschine 12 vorzugsweise dadurch gekennzeichnet, daß sie einen einheitlichen bzw. gleichmäßigen Speicheradressenraum aufweist. Bei der bevorzugten Ausführungsform enthalten Programminstruktionen, die in dem Speicher 34 gespeichert sind, selektiv Rekonfigurationsanweisungen, die an die DRPU 32 gerichtet sind. Nimmt man nun Bezug auf die Fig. 3A, so ist ein beispielhaftes Programmlisting 50 einschließlich der Rekonfigurationsanweisungen gezeigt. Wie in der Fig. 3A gezeigt ist, beinhaltet das beispielhafte Programmlisting 50 einen Satz von Außenschleifenabschnitten 52, einen ersten Innenschleifenabschnitt 54, einen zweiten Innenschleifenabschnitt 55, einen dritten Innenschleifenabschnitt 56, einen vierten Innenschleifenabschnitt 57 und einen fünften Innenschleifenabschnitt 58. Fachleute werden leicht erkennen, daß der Ausdruck "Innenschleife" bzw. "Inner-Loop" auf einen iterativen Abschnitt eines Programms Bezug nimmt, der für die Durchführung eines bestimmten Satzes von dazu in bezug stehenden Operationen verantwortlich ist, und der Term "Außenschleife" bzw. "Outer-Loop" nimmt auf jene Abschnitte eines Programms Bezug, die hauptsächlich für die Durchführung von Allzweckoperationen und/oder für das Übertragen einer Steuerung von einem Innenschleifenabschnitt zu einem anderen verantwortlich ist. Im allgemeinen führen Innenschleifenabschnitte 54, 55, 56, 57, 58 eines Programmes Operationen bezüglich potentiell großer Datensätze durch. Bei einer Bildverarbeitungsanwendung kann z. B. der erste Innenschleifenabschnitt 54 eine Farbformatumwandlungsoperation bezüglich der Bilddaten durchführen und die zweiten bis fünften Innenschleifenabschnitte 55, 56, 57, 58 können ein lineares Filtern, Falten, Mustersuchen und Komprimierungsoperationen durchführen. Fachleute werden erkennen können, daß eine aneinanderliegende Abfolge von Innenschleifenabschnitten 55, 56, 57, 58 als eine Software-Pipeline bzw. ein Software-Fließband gedacht werden kann. Jeder Außenschleifenabschnitt 52 wäre für den Daten-I/O und/oder für die Anweisung des Transfers bzw. der Übertragung von Daten und der Steuerung von dem ersten Innenschleifenabschnitt 54 zu dem zweiten Innenschleifen-Abschnitt 55 verantwortlich. Fachleute werden zusätzlich erkennen, daß ein gegebener Innenschleifenabschnitt 54, 55, 56, 57, 58 eine oder mehrere Rekonfigurationsanweisungen enthalten kann. Im allgemeinen werden für jedes gegebene Programm die Außenschleifenabschnitte 52 oder das Programmlisting 50 eine Vielzahl von Allzweck-Instruktionstypen beinhalten, während die Innenschleifenabschnitte 54, 56 des Programmlistings 50 aus relativ wenigen Instruktionstypen bestehen, die verwendet werden, um einen spezifischen Satz von Operationen durchzuführen.

Bei dem beispielhaften Programmlisting 50 erscheint eine erste Rekonfigurationsanweisung am Beginn des ersten Innenschleifenabschnittes 54 und eine zweite Rekonfigurationsanweisung erscheint an dem Ende des ersten Innenschleifenabschnittes 54. In ähnlicher Weise erscheint eine dritte Rekonfigurationsanweisung am Beginn des zweiten Innenschleifenabschnittes 55; eine vierte Rekonfigurationsanweisung erscheint am Beginn des dritten Innenschleifenabschnittes 56; eine fünfte Rekonfigurationsanweisung erscheint am Beginn des vierten Innenschleifenabschnittes 57; und eine sechste und siebte Rekonfigurationsanweisung erscheint am Beginn und am Ende des fünften Innenschleifenabschnittes 58, und zwar jeweils. Jede Rekonfigurationsanweisung nimmt vorzugsweise auf einen Konfigurationsdatensatz Bezug, der eine interne DRPU-Hardware-Organisation spezifiziert, die auf die Implementation bzw. Realisation einer bestimmten Instruktionssatz-Architektur bzw. "Instruction Set Architecture" (ISA) ausgerichtet ist und dafür optimiert ist. Eine ISA ist ein primitiver Satz oder Kernsatz von Instruktionen, der verwendet werden kann, um einen Computer zu programmieren. Eine ISA definiert Instruktionssformate, Operationscodes bzw. Opcodes, Datenformate, Adressierungsmoden, Ausführungssteuermarken bzw. Ausführungssteuerflags und über das Programm zugreifbare Register. Fachleute werden erkennen, daß dies der konventionellen Definition einer ISA entspricht. Bei der vorliegenden Erfindung

kann jede DRPU 32 einer S-Maschine schnell in Echtzeit konfiguriert werden, um direkt mehrere ISAs durch die Verwendung eines einzigen Konfigurationsdatensatzes für jede gewünschte ISA zu realisieren. Das heißt, jede ISA wird mit einer einzigen internen DRPU-Hardware-Organisation realisiert bzw. implementiert, wie dies durch einen entsprechenden Konfigurationsdatensatz spezifiziert ist. Somit entsprechen bei der vorliegenden Erfindung die ersten bis fünften Innenschleifenabschnitte 54, 55, 56, 57, 58 jeweils einer einzigen ISA, nämlich ISA jeweils 1, 2, 3, 4 und k. Fachleute werden erkennen, daß jede aufeinanderfolgende bzw. sukzessive ISA nicht einzig zu sein braucht. Somit kann ISA k ISA 1, 2, 3, 4 oder jede davon unterschiedliche ISA sein. Der Satz von Außenschleifenabschnitten 52 entspricht ebenso einer einzigen ISA, nämlich ISA 0. Bei der bevorzugten Ausführungsform kann während der Programmausführung die Auswahl sukzessiver Rekonfigurationsanweisungen von den Daten abhängen. Nach der Auswahl einer gegebenen Rekonfigurationsanweisung werden die Programmstrukturen aufeinanderfolgend gemäß einer entsprechenden ISA über eine einzige DRPU-Hardware-Konfiguration ausgeführt, wie dies durch einen entsprechenden Konfigurationsdatensatz spezifiziert ist.

Bei der vorliegenden Erfindung kann eine gegebene ISA als eine Innenschleifen-ISA oder eine Außenschleifen-ISA gemäß der Anzahl und Typen von Instruktionen, die sie enthält, kategorisiert werden. Eine ISA, die mehrere Instruktionen beinhaltet, und das ist nützlich, um Allzweck-Operationen durchzuführen, ist eine Außenschleifen-ISA, während eine ISA, die aus relativ wenigen Instruktionen besteht, und das ist daraufhin ausgerichtet, spezifische Typen von Operationen durchzuführen, ist eine Innenschleifen-ISA. Weil eine Außenschleifen-ISA auf die Durchführung von Allzweck-Operationen gerichtet ist, ist eine Außenschleifen-ISA besonders nützlich, wenn eine aufeinander abfolgende bzw. sequentielle Ausführung von Programmstrukturen erwünscht ist. Die Leistungsfähigkeit einer Außenschleifen-ISA bezüglich der Ausführung wird vorzugsweise in Termen der Taktzyklen pro ausgeführter Instruktion charakterisiert. Im Gegensatz dazu ist eine Innenschleifen-ISA, da eine Innenschleifen-ISA auf die Ausführung spezieller Typen von Operationen gerichtet ist, am nützlichsten, wenn eine Ausführung von Parallel-Programmstrukturen wünschenswert ist. Die Leistungsfähigkeit einer Innenschleifen-ISA wird vorzugsweise in Termen von ausgeführten Instruktionen pro Taktzyklus oder Rechenergebnis, das pro Taktzyklus erzeugt wird, charakterisiert.

Fachleute werden erkennen, daß die vorhergehende Diskussion einer Ausführung von sequentiellen Programmstrukturen und einer Ausführung von parallelen Programmstrukturen die Ausführung von Programmstrukturen innerhalb einer einzigen DRPU 32 betrifft. Das Vorliegen von mehreren S-Maschinen 12 in dem System 10 erleichtert die parallele Ausführung von mehreren Programmstrukturensequenzen zu jeder gegebenen Zeit, und zwar wo jede Programmstrukturensequenz durch eine gegebene DRPU 32 ausgeführt wird. Jede DRPU 32 ist so konfiguriert, daß sie eine parallele bzw. serielle Hardware aufweist, um eine bestimmte Innenschleifen-ISA bzw. Außenschleifen-ISA zu einer bestimmten Zeit zu implementieren. Die interne Hardwarekonfiguration einer gegebenen DRPU 32 ändert sich mit der Zeit gemäß der Auswahl von einer oder mehreren Rekonfigurationsanweisungen, die innerhalb einer Sequenz von Programmstrukturen, die ausgeführt werden, eingebettet sind.

Bei der bevorzugten Ausführungsform werden jede ISA und ihre entsprechende interne DRPU-Hardware-Organisation ausgestaltet, um optimale Rechenleistungsfähigkeit für eine bestimmte Klasse von Rechenproblemen relativ zu einem Satz von verfügbaren rekonfigurierbaren Hardware-Systemeinheiten bereitzustellen. Wie zuvor erwähnt wurde und wie detaillierter weiter unten beschrieben werden wird, wird eine interne DRPU-Hardware-Organisation, die einer Außenschleifen-ISA entspricht, vorzugsweise für die Ausführung sequentieller Programmstrukturen optimiert und eine interne DRPU-Hardware-Organisation, die einer Innenschleifen-ISA entspricht, wird vorzugsweise für eine Ausführung paralleler Programmstrukturen optimiert. Eine beispielhafte Allzweck-Außenschleifen-ISA ist Anhang A zu entnehmen und eine beispielhafte Innenschleifen-ISA, die auf die Faltung gerichtet ist, ist dem Anhang B zu entnehmen.

Mit Ausnahme jeder Rekonfigurationsanweisung, umfaßt das beispielhafte Programmlisting 50 der Fig. 3A vorzugsweise herkömmliche Sprachbefehle hohen Niveaus, z. B. Befehle, die in Übereinstimmung mit der C-Programmiersprache beschrieben sind. Fachleute werden erkennen, daß der Einschluß von einer oder mehreren Konfigurationsanweisungen in eine Sequenz von Programmstrukturen einen Kompiler erfordert, der modifiziert ist bzw. geändert ist, um den Rekonfigurationsanweisungen Rechnung zu tragen. Nimmt man nun auf die Fig. 3B Bezug, so ist ein Flußdiagramm nach dem Stand der Technik für Kompilierungsoperationen gezeigt, die während der Kompilierung einer Sequenz bzw. Abfolge von Programmstrukturen durchgeführt werden. Hierin entsprechen die Kompilierungsoperationen gemäß dem Stand der Technik im allgemeinen jenen, die durch den GNU-C-Kompiler (GCC) durchgeführt werden, der von der Free Software Foundation (Cambridge, MA, USA) hergestellt werden. Fachleute werden erkennen, daß die Kompilierungsoperationen gemäß dem Stand der Technik, die weiter unten beschrieben sind, leicht für andere Kompiler verallgemeinert werden können. Die Kompilierungsoperationen nach dem Stand der Technik beginnen im Schritt 500, wobei das Kompiler-Vorderende einen nächsten Befehl hohen Niveaus aus einer Abfolge von Programmstrukturen auswählt. Danach erzeugt das Kompiler-Vorderende einen Code mittleren Niveaus, der dem gewählten Befehl hohen Niveaus im Schritt 502 entspricht, was in dem Fall des GCC dem Registertransferniveau- bzw. "Register Transfer Level" (RTL) -Befehlen entspricht. Folgt man nun Schritt 502, so bestimmt das Vorderende des Kompilers, ob ein anderer Befehl hohen Niveaus im Schritt 504 eine Beachtung erfordert. Falls dem so ist, so kehrt das bevorzugte Verfahren zu dem Schritt 500 zurück.

Im Schritt 504 bestimmt das Kompiler-Vorderende, daß kein anderer Befehls hohen Niveaus eine Beachtung erfordert, das Kompiler-Hinterende führt als nächstes herkömmliche Registerzuweisungsoperationen im Schritt 506 durch. Nach dem Schritt 506 wählt das Kompiler-Hinterende einen nächsten RTL-Befehl, der innerhalb einer gegenwärtigen RTL-Befehlsgruppe im Schritt 508 zu beachten ist. Das Kompiler-Hinterende bestimmt dann, ob eine Regel im Schritt 510 existiert, die eine Art und Weise spezifiziert, in der die gegenwärtige RTL-Befehlsgruppe in einen Satz von Assemblersprachenbefehlen übersetzt werden kann. Falls eine derartige

Regel nicht existiert, kehrt das bevorzugte Verfahren zum Schritt 508 zurück, um einen anderen RTL-Befehl auszuwählen, um ihn in die gegenwärtige RTL-Befehlsgruppe einzuschließen. Falls eine Regel existiert, die der gegenwärtigen RTL-Befehlsgruppe entspricht, erzeugt das Kompiler-Hinterende einen Satz von Assemblersprachenbefehlen gemäß der Regel in dem Schritt 512. Nachfolgend zum Schritt 512 bestimmt das Kompiler-Hinterende, ob ein nächster RTL-Befehl eine Beachtung erfordert, und zwar im Zusammenhang mit einer nächsten RTL-Befehlsgruppe. Falls dem so ist, kehrt das bevorzugte Verfahren zum Schritt 508 zurück; ansonsten endet das bevorzugte Verfahren.

Die vorliegende Erfindung beinhaltet vorzugsweise einen Kompiler für dynamisch rekonfigurierbares Rechnen. Nimmt man nun Bezug auf die Fig. 3C und 3D, so ist ein Flußdiagramm für bevorzugte Kompilierungsoperationen gezeigt, die von einem Kompiler für eine dynamisch rekonfigurierbare Berechnung durchgeführt werden. Die bevorzugten Kompilierungsoperationen beginnen beim Schritt 600 mit dem Vorderende des Kompilers für dynamisch rekonfigurierbares Berechnen, wobei ein nächster Befehl hohen Niveaus innerhalb einer Sequenz von Programminstruktionen ausgewählt wird. Als nächstes bestimmt das Vorderende des Kompilers für dynamisch rekonfigurierbares Berechnen, ob der gewählte Befehl hohen Niveaus eine Rekonfigurationsanweisung ist, und zwar im Schritt 602. Falls dem so ist, erzeugt das Vorderende des Kompilers für dynamisch rekonfigurierbares Rechnen einen RTL-Rekonfigurationsbefehl im Schritt 604, nachdem das bevorzugte Verfahren zu dem Schritt 600 zurückkehrt. Bei der bevorzugten Ausführungsform handelt es sich bei dem RTL-Rekonfigurationsbefehl um einen Nicht-Standard-RTL-Befehl, der eine ISA-Identifikation bzw. ISA-Kennzeichnung beinhaltet. Falls im Schritt 602 der gewählte Programmbebefhl hohen Niveaus nicht eine Rekonfigurationsanweisung ist, erzeugt das Vorderende des Kompilers für dynamisch rekonfigurierbares Rechnen als nächstes einen Satz von RTL-Befehlen auf eine konventionelle Art und Weise, und zwar im Schritt 606. Nach dem Schritt 606 bestimmt das Vorderende des Kompilers für dynamisch rekonfigurierbares Rechnen, ob ein anderer Befehl hohen Niveaus eine Beachtung erfordert, und zwar im Schritt 608. Falls dem so ist, kehrt das bevorzugte Verfahren zum Schritt 600 zurück; ist dem nicht so, fährt das bevorzugte Verfahren zu dem Schritt 610 fort, um Operationen bezüglich des hinteren Endes auszulösen bzw. zu beginnen.

Im Schritt 610 führt das Hinterende des Kompilers für dynamisch rekonfigurierbares Rechnen Registerzuordnungsoperationen durch. Bei der bevorzugten Ausführungsform der vorliegenden Erfindung ist jede ISA derartig definiert, daß die Registerarchitektur von einer ISA zu einer anderen konsistent ist; deshalb werden die Registerzuordnungsoperationen auf eine konventionelle Art und Weise durchgeführt. Fachleute werden erkennen, daß im allgemeinen eine konsistente Registerarchitektur von einer ISA zur anderen kein absolutes Erfordernis ist. Als nächstes wählt das Hinterende des Kompilers für dynamisch rekonfigurierbares Rechnen einen nächsten RTL-Befehl innerhalb einer gegenwärtig betrachteten RTL-Befehlsgruppe im Schritt 612. Das Hinterende des Kompilers für dynamisch rekonfigurierbares Rechnen bestimmt dann im Schritt 614, ob der gewählte RTL-Befehl eine RTL-Rekonfigurationsbefehl ist. Falls der gewählte RTL-Befehl kein RTL-Rekonfigurationsbefehl ist, bestimmt das Hinterende des Kompilers für dynamisch rekonfigurierbares Rechnen im Schritt 618, ob eine Regel für die gegenwärtig betrachtete RTL-Befehlsgruppe existiert. Falls nicht, kehrt das bevorzugte Verfahren zum Schritt 612 zurück, um einen nächsten RTL-Befehl auszuwählen, der in die gegenwärtig betrachtete RTL-Befehlsgruppe eingeschlossen werden soll. Für den Fall, daß eine Regel für die gegenwärtig betrachtete RTL-Befehlsgruppe im Schritt 618 existiert, erzeugt das Hinterende des Kompilers für dynamisch rekonfigurierbares Rechnen als nächstes einen Satz von Assemblersprachbefehlen, die der gegenwärtig betrachteten RTL-Befehlsgruppe gemäß dieser Regel entspricht, und zwar im Schritt 620. Nachfolgend zum Schritt 620 bestimmt das Hinterende des Kompilers für dynamisch rekonfigurierbares Rechnen, ob ein anderer RTL-Befehl eine Beachtung innerhalb des Zusammenhangs einer nächsten RTL-Befehlsgruppe erfordert, und zwar im Schritt 622. Falls dem so ist, kehrt das bevorzugte Verfahren zum Schritt 612 zurück, falls dem nicht so ist, endet das bevorzugte Verfahren.

Im Schritt 614 handelt es sich bei dem gewählten RTL-Befehl um einen RTL-Rekonfigurationsbefehl, das Hinterende des Kompilers für dynamisch rekonfigurierbares Rechnen wählt einen Regelsatz, der der ISA-Identifikation innerhalb des RTL-Rekonfigurationsbefehls entspricht, und zwar im Schritt 616. Bei der vorliegenden Erfindung existiert vorzugsweise ein einziger Regelsatz für jede ISA. Jeder Regelsatz liefert deshalb eine oder mehrere Regeln zur Umwandlung von Gruppen von RTL-Befehlen in Assemblersprachbefehle in Übereinstimmung mit einer bestimmten ISA. Nachfolgend zum Schritt 616 läuft das bevorzugte Verfahren weiter zum Schritt 618. Der Regelsatz, der einer jeden gegebenen ISA entspricht, beinhaltet vorzugsweise eine Regel, um den RTL-Rekonfigurationsbefehl in einen Satz von Assemblersprachbefehlen zu übersetzen, die einen Software-Interrupt erzeugen, der in der Ausführung einer Rekonfigurations-Handhabungseinrichtung resultiert, wie dies im folgenden detailliert beschrieben werden wird.

In der oben beschriebenen Art und Weise erzeugt der Kompiler für dynamisch rekonfigurierbares Rechnen selektiv und automatisch Assemblersprachbefehle, und zwar in Übereinstimmung mit mehreren ISAs während der Kompilierungsoperationen. Mit anderen Worten kompiliert während des Kompilierungsprozesses der Kompiler für dynamisch rekonfigurierbares Rechnen einen einzigen Satz von Programminstruktionen gemäß einer variablen ISA. Bei dem Kompiler für dynamisch rekonfigurierbares Rechnen handelt es sich vorzugsweise um einen herkömmlichen Kompiler, der modifiziert ist, um die bevorzugten Kompilierungsoperationen durchzuführen, die oben unter Bezugnahme auf die Fig. 3C und 3D beschrieben sind. Fachleute werden erkennen, daß, obwohl die erforderlichen Modifikationen bzw. Änderungen nicht komplex sind, derartige Modifikationen im Hinblick auf sowohl Kompilierungstechniken nach dem Stand der Technik als auch rekonfigurierbare Rechen-techniken nach dem Stand der Technik nicht offensichtlich sind.

Nimmt man nun Bezug auf die Fig. 4, so ist ein Blockschaltbild bzw. ein Blockdiagramm einer bevorzugten Ausführungsform einer dynamischen rekonfigurierbaren Verarbeitungseinheit 32 gezeigt. Die DRPU 32 umfaßt eine Instruktionsanforderungseinheit (IFU) 60, eine Datenoperationseinheit (DOU) 62 und eine Adressenopera-

tionseinheit (AOU) 64. Sowohl die IFU 60 als auch die DOU 62 und die AOU 64 umfassen einen Zeitsteuereingang, der mit der ersten Zeitsteuersignalleitung 40 verbunden ist. Die IFU 60 umfaßt einen Speichersteuerausgang, der mit der Speichersteuerleitung 42 verbunden ist, einen Dateneingang, der mit der speicher-I/O-Leitung 46 verbunden ist, und einen bidirektionalen Steuerport, der mit der externen Steuerleitung 48 verbunden ist. Die IFU 60 umfaßt zusätzlich einen ersten Steuerausgang, der mit einem ersten Steuereingang der DOU 62 über eine erste Steuerleitung 70 verbunden ist, und einen zweiten Steuerausgang, der mit einem zweiten Steuereingang der AOU 64 über eine zweite Steuerleitung 72 verbunden ist. Die IFU 60 umfaßt ebenso einen dritten Steuerausgang, der mit einem zweiten Steuereingang der DOU 62 und einem zweiten Steuereingang der AOU 64 über eine dritte Steuerleitung 74 verbunden ist. Die DOU 62 und die AOU 64 umfassen jeweils einen bidirektionalen Datenport, der mit der speicher-I/O-Leitung 46 verbunden ist. Schließlich umfaßt die AOU 64 einen Adressenausgang, der den Adressenausgang der DRPU ausbildet.

Die DRPU 32 wird vorzugsweise realisiert, indem eine rekonfigurierbare oder reprogrammierbare Logikvorrichtung, wie z. B. eine FPGA, wie z. B. eine Xilinx XC4013 (Xilinx, Inc., San Jose, CA, USA) oder eine AT&T ORCATM 1C07 (AT&T Microelectronics, Allentown, PA, USA) verwendet wird. Vorzugsweise stellt die reprogrammierbare Logikvorrichtung eine Vielzahl von folgendem bereit: 1) selektiv reprogrammierbare Logikblöcke oder konfigurierbare Logikblöcke (CLBs); 2) selektiv reprogrammierbare I/O-Blöcke (IOBs); 3) selektiv reprogrammierbare Verbindungsstrukturen; 4) Datenspeicher-Systemeinheiten; 5) Dreizustands-Puffer-Systemeinheiten; und 6) Funktionsfähigkeiten einer fest verdrahteten Logik. Jede CLB beinhaltet vorzugsweise eine selektiv rekonfigurierbare Schaltung zur Erzeugung von Logikfunktionen, Speicherdaten und Wegeermittlungssignalen bzw. Leitwegsignalen. Fachleute werden erkennen, daß eine rekonfigurierbare Datenspeicher-Schaltung auch in einer oder mehreren Datenspeicherblöcken (DSBs) beinhaltet sein können, die von dem Satz von CLBs getrennt sind, und zwar in Abhängigkeit von der exakten Ausgestaltung der rekonfigurierbaren Logikvorrichtung, die verwendet wird. Hier befindet sich die rekonfigurierbare Datenspeicherschaltung, die innerhalb einer FPGA ist, innerhalb der CLBs; d. h. die Gegenwart von DSBs wird nicht angenommen. Fachleute werden leicht erkennen, daß eine oder mehrere Elemente, die hierin beschrieben sind, die eine CLB-basierte rekonfigurierbare Datenspeicherschaltung verwenden, eine DSB-basierte Schaltung für den Fall verwenden könnten, daß DSBs vorhanden sind. Jede IOB beinhaltet vorzugsweise eine selektiv rekonfigurierbare Schaltung, um Daten zwischen CLBs und einem FPGA-Ausgangspunkt zu übertragen. Ein Konfigurationsdatensatz legt eine DRPU-Hardware-Konfiguration oder -Organisation fest, indem Funktionen spezifiziert werden, die innerhalb von CLBs durchgeführt werden, sowie Verbindungen spezifiziert werden, und zwar wie folgt: 1) innerhalb CLBs; 2) zwischen CLBs; 3) innerhalb IOBs; 4) zwischen IOBs; und 5) zwischen CLBs und IOBs. Fachleute werden erkennen, daß über einen Konfigurationsdatensatz die Anzahl von Bits sowohl in der Speichersteuerleitung 42 als auch in der Adreßleitung 44, der speicher-I/O-Leitung 46 und der externen Steuerleitung 48 rekonfigurierbar ist. Vorzugsweise werden Konfigurationsdatensätze in einem oder mehreren S-Maschinenspeichern 34 innerhalb des Systems 10 gespeichert. Fachleute werden erkennen, daß die DRPU 32 nicht auf eine FPGA-basierte Implementation bzw. Realisierung beschränkt ist. Zum Beispiel könnte die DRPU 32 als eine RAM-basierte Zustandsmaschine verwirklicht werden, die möglicherweise eine oder mehrere Nachschlag- bzw. Verweistabellen enthält. Alternativ könnte die DRPU 32 realisiert werden, indem eine komplexe programmierbare Logikvorrichtung bzw. eine "Complex Programmable Logic Device" (CPLD) verwendet wird. Jedoch werden Fachleute erkennen, daß einige der S-Maschinen 12 des Systems 10 DRPUs 32 enthalten können, die nicht rekonfigurierbar sind.

Bei der bevorzugten Ausführungsform sind sowohl die IFU 60 als auch die DOU 62 und die AOU 64 dynamisch rekonfigurierbar. Somit kann ihre interne Hardware-Konfiguration selektiv während der Programmierung geändert werden. Die IFU 60 verwaltet Instruktionsanweisungs- und Decodieroperationen, Speicherzugriffsoperationen, DRPU-Rekonfigurationsoperationen und gibt Steuersignale zu der DOU 62 und der AOU 64 aus, um die Instruktionsausübung zu erleichtern. Die DOU 62 führt Operationen aus, die eine Datenberechnung mit einschließen bzw. mit sich bringen und die AOU 64 führt Operationen aus, die eine Adressenberechnung mit sich bringen. Die interne Struktur und der Betrieb sowohl der IFU 60 als auch der DOU 62 und der AOU 64 wird nun detailliert beschrieben.

Nimmt man Bezug auf die Fig. 5, so ist ein Blockdiagramm bzw. ein Blockschaltbild einer bevorzugten Ausführungsform der Instruktionsabrufeinheit 60 bzw. "Instruction Fetch Unit" 60 gezeigt. Die IFU 60 umfaßt eine Instruktionszustandsfolgesteuereinheit bzw. einen "Instruction State Sequencer" (ISS) 100, einen Architekturbeschreibungsspeicher 101, eine Speicherzugriffslogik 102, eine Rekonfigurationslogik 104, eine Interruptlogik 106, eine Abrufsteuereinheit 108, einen Instruktionspuffer 110, eine Dekodersteuereinheit 112, einen Instruktionsdecoder 114, einen Operationscode-Speicherregistersatz 116, einen Registerfile (RF)-Adressenregistersatz 118, einen Konstantenregistersatz 120 und einen Prozeßsteuerregistersatz 122. Die ISS 100 umfaßt einen ersten bzw. einen zweiten Steuerausgang, der den ersten bzw. zweiten Steuerausgang der IFU ausbildet, und einen Zeitsteuereingang, der den Zeitsteuereingang der IFU ausbildet. Die ISS 100 umfaßt ebenso einen Abruf-/Decodiersteuerausgang, der mit einem Steuereingang der Abrufsteuereinheit 108 und einem Steuereingang der Decodiersteuereinheit 112 über eine Abruf-/Decodiersteuerleitung 130 verbunden ist. Die ISS 100 weist zusätzlich einen bidirektionalen Steuerport auf, der mit einem ersten bidirektionalen Steuerport sowohl der Speicherzugriffslogik 102 als auch der Rekonfigurationslogik 104 und der Interruptlogik 106 über eine bidirektionale Steuerleitung 132 verbunden ist. Die ISS 100 umfaßt ebenso einen Operationscodeeingang, der mit einem Ausgang des Operationscodespeicherregistersatzes 116 über eine Operationscodeleitung 142 verbunden ist. Schließlich umfaßt die ISS 100 einen bidirektionalen Datenport, der mit einem bidirektionalen Datenport des Prozeßsteuerregistersatzes 122 über eine Prozeßdatenleitung 144 verbunden ist.

Sowohl die Speicherzugriffslogik 102 als auch die Rekonfigurationslogik 104 und die Interruptlogik 106 umfassen einen zweiten bidirektionalen Steuerport, der mit der externen Steuerleitung 48 verbunden ist. Die

Speicherzugriffslogik 102, die Rekonfigurationslogik 104 und die Interruptlogik 106 umfassen zusätzlich jeweils einen Dateneingang, der mit einem Datenausgang des Architekturbeschreibungsspeichers 101 über eine Implementationssteuerleitung 131 bzw. Realisierungssteuerleitung 131 verbunden ist. Die Speicherzugriffslogik 102 umfaßt zusätzlich einen Steuerausgang, der den Speichersteuerausgang der IFU ausbildet. Und die Interruptlogik 106 umfaßt einen Ausgang, der mit der Prozeßdatenleitung 144 verbunden ist. Der Instruktionsspuffer 110 umfaßt einen Dateneingang, der den Dateneingang der IFU ausbildet, einen Steuereingang, der mit einem Steuerausgang der Abrufsteuereinheit 108 über eine Abrufsteuerleitung 134 verbunden ist, und einen Ausgang, der mit einem Eingang des Instruktionsdecoders 114 über eine Instruktionsleitung 136 verbunden ist. Der Instruktionsdecoder 114 umfaßt einen Steuereingang, der mit einem Steuerausgang der Decodiersteuereinheit 112 über eine Decodiersteuerleitung 138 verbunden ist, und einen Ausgang, der über eine Decodierinstruktionleitung 140 mit 1) einem Eingang des Operationscode-Speicherregistersatzes 116; 2) einem Eingang des RF-Adressenregistersatzes 118; und 3) einem Eingang des Konstantenregistersatzes 120 verbunden ist. Der RF-Adressenregistersatz 118 und der Konstantenregistersatz 120 umfassen jeweils einen Ausgang, die zusammen den dritten Steuerausgang 74 der IFU ausbilden.

Der Architekturbeschreibungsspeicher 101 speichert Architekturbeschreibungssignale, die die gegenwärtige DRPU-Konfiguration kennzeichnen. Vorzugsweise beinhalten die Architektspezifikationssignale 1) einen Bezug bzw. eine Referenz zu einem Ausgangskonfigurationsdatensatz bzw. Default-Konfigurationsdatensatz; 2) einen Bezug bzw. eine Referenz zu einer Liste von möglichen Konfigurationsdatensätzen; 3) einen Bezug bzw. eine Referenz zu einem Konfigurationsdatensatz, der der gegenwärtig betrachteten ISA entspricht, d. h. einen Bezug zu dem Konfigurationsdatensatz, der die gegenwärtige DRPU-Konfiguration festlegt; 4) eine Verbindungsadressenliste, die eine oder mehrere Verbindungs-I/O-Einheiten 304 innerhalb der T-Maschine 14 identifiziert, die der S-Maschine 12 zugeordnet ist, in der sich die IFU 60 befindet, wie detailliert weiter unten unter Bezugnahme auf die Fig. 13 beschrieben werden wird; 5) einen Satz von Interrupt-Antwortsignalen, die eine Interruptsuchzeit bzw. Interruptverzögerungszeit und eine Interrupt-Präzisionsinformation, die festlegt, wie die IFU 60 auf die Interrupt antwortet, spezifizieren; und 6) eine Speicherzugriffskonstante, die eine atomare Speicheradresseninkrementierung festlegt. Bei der bevorzugten Ausführungsform realisiert bzw. implementiert jeder Konfigurationsdatensatz z den Architekturbeschreibungsspeicher 101 als einen Satz von CLBs, der als ein Nur-Lese-Speicher bzw. "Read Only Memory" (ROM) konfiguriert ist. Die Architektspezifikationssignale, die den Inhalt des Architekturbeschreibungsspeichers 101 festlegen, sind vorzugsweise in jedem Konfigurationsdatensatz enthalten. Da jeder Konfigurationsdatensatz einer bestimmten ISA entspricht, variiert der Inhalt des Architekturbeschreibungsspeichers 101 gemäß der ISA, die gegenwärtig betrachtet wird. Für eine gegebene ISA wird der Programmzugriff auf den Inhalt des Architekturbeschreibungsspeichers 101 vorzugsweise erleichtert, indem eine Speicherleseinstruktion in die ISA eingeschlossen wird bzw. mit aufgenommen wird. Dies ermöglicht es, daß ein Programm Informationen über die gegenwärtige DRPU-Konfiguration während der Programmausführung wiederfindet.

Bei der vorliegenden Erfindung handelt es sich bei der Rekonfigurationslogik 104 um eine Zustandsmaschine, die eine Abfolge von Rekonfigurationsoperationen steuert, die die Rekonfiguration der DRPU 32 gemäß einem Konfigurationsdatensatz erleichtert. Vorzugsweise löst die Rekonfigurationslogik 104 die Rekonfigurationsoperationen nach dem Empfang eines Rekonfigurationssignals aus. Wie detailliert weiter unten beschrieben werden wird, wird das Rekonfigurationssignal durch die Interruptlogik 106 in Antwort auf einen Rekonfigurationsinterrupt erzeugt, der auf der externen Steuerleitung 48 empfangen wird, oder es wird durch die ISS 100 in Antwort auf eine Rekonfigurationsanweisung, die in einem Programm eingebaut ist, erzeugt. Die Rekonfigurationsoperationen stellen eine anfängliche DRPU-Konfiguration bereit, die einer Einschalt-/Reset-Bedingung folgt, die den Ausgangs-Konfigurationsdatensatz bzw. Default-Konfigurationsdatensatz verwendet, auf den der Architekturbeschreibungsspeicher 101 verweist. Die Rekonfigurationsoperationen stellen ebenso eine selektive DRPU-Rekonfiguration bereit, nachdem die anfängliche DRPU-Konfiguration erstellt worden ist. Nach der Vollendung der Rekonfigurationsoperationen gibt die Rekonfigurationslogik 104 ein Vollendungssignal ab. Bei der bevorzugten Ausführungsform handelt es sich bei der Rekonfigurationslogik 104 um eine nicht-rekonfigurierbare Logik, die das Laden von Konfigurationsdatensätzen in die reprogrammierbare Logikvorrichtung selbst steuert, und somit wird die Abfolge bzw. Sequenz von Rekonfigurationsoperationen durch den Hersteller der reprogrammierbaren Logikvorrichtung festgelegt. Die Rekonfigurationsoperationen werden nun für die Fachleute bekannt sein.

Jede DRPU-Konfiguration ist vorzugsweise durch einen Konfigurationsdatensatz gegeben, der eine bestimmte Hardware-Organisation festlegt, die auf die Implementation bzw. Realisierung einer entsprechenden ISA ausgerichtet ist bzw. zugeordnet ist. Bei der bevorzugten Ausführungsform beinhaltet die IFU 60 jedes der Elemente, auf die oben hingewiesen wurde, und zwar unabhängig von der DRPU-Konfiguration. Bei einem Basisniveau bzw. bei einer Grundebene ist die Funktionalität bzw. Funktionstüchtigkeit, die durch jedes Element innerhalb der IFU 60 bereitgestellt wird, unabhängig von der gegenwärtig betrachteten ISA. Jedoch kann bei der bevorzugten Ausführungsform die detaillierte Struktur und Funktionstüchtigkeit von einem oder mehreren Elementen der IFU 60 sich verändern, und zwar aufgrund der Natur der ISA, für die sie konfiguriert worden ist. Bei der bevorzugten Ausführungsform bleibt die Struktur und Funktionalität des Architekturbeschreibungsspeichers 101 und der Rekonfigurationslogik 104 vorzugsweise von einer DRPU-Konfiguration zu einer anderen konstant. Die Struktur und Funktionstüchtigkeit bzw. Funktionalität der anderen Elemente der IFU 60 und die Art und Weise, in der sie gemäß dem Typ der ISA variiieren, wird nun im Detail beschrieben.

Der Prozeßsteuerregistersatz 122 speichert Signale und Daten, die durch die ISS 100 während der Ausführung von Instruktionen verwendet wird. Bei der bevorzugten Ausführungsform umfaßt der Prozeßsteuerregistersatz 122 ein Register, um ein Prozeßsteuerwort zu speichern, ein Register, um einen Interruptvektor zu speichern, und ein Register, um einen Bezug zu einem Konfigurationsdatensatz zu speichern. Das Prozeßsteuerwort

beinhaltet vorzugsweise eine Vielzahl von Bedingungsflags, die wahlweise gesetzt und rückgesetzt werden können, und zwar in Abhängigkeit von Bedingungen, die während der Instruktionsausführung auftreten. Das Prozeßsteuerwort beinhaltet zusätzlich eine Vielzahl von Übergangssteuersignalen, die eine oder mehrere Arten und Weisen festlegen, in denen Interrupts abgearbeitet werden können, wie im folgenden detaillierter beschrieben werden wird. Bei der bevorzugten Ausführungsform ist der Prozeßsteuerregistersatz 122 als ein Satz von CLBs realisiert, der für Datenspeicherung und für Gattersteuerlogik bzw. logische Torschaltungen konfiguriert ist.

Die ISS 100 ist vorzugsweise eine Zustandsmaschine, die den Betrieb der Abrufsteuereinheit 108, der Decodiersteuereinheit 112, der DOU 62 und der AOU 64 steuert und Speicherlese- und Speicherschreibsignale zu der Speicherzugriffslogik 102 ausgibt, um die Instruktionsausführung zu erleichtern. Nimmt man nun Bezug auf Fig. 6, so ist ein Zustandsdiagramm gezeigt, das einen bevorzugten Satz von Zuständen, die durch die ISS 100 unterstützt werden, zeigt. Nachfolgend zu einer Einschalt- oder Resetbedingung oder unmittelbar, nachdem eine Rekonfiguration aufgetreten ist, beginnt die ISS 100 mit einer Operation im Zustand P. In Antwort auf das Vollendungssignal, das durch die Rekonfigurationslogik 104 ausgegeben wird, schreitet die ISS 100 zum Zustand S fort, indem die ISS 100 Programmzustandsinformationen initialisiert wiederhergestellt, und zwar in dem Fall, daß eine Einschalt-/Resetbedingung bzw. eine Rekonfiguration aufgetreten ist. Die ISS 100 schreitet zum Zustand F fort, indem Instruktionsabrufoperationen durchgeführt werden. Bei den Instruktionsabrufoperationen gibt die ISS 100 ein Speicherlesesignal zu der Speicherzugriffslogik 102 aus, gibt ein Abrufsignal zu der Abrufsteuereinheit 108 aus und gibt ein Inkrementesignal zu der AOU 64 aus, um ein Programmadressenregister bezüglich einer nächsten Instruktion bzw. ein "Next Instruction Programm Address Register" (NIPAR) 232 zu inkrementieren, wie unten detaillierter unter Bezugnahme auf die Fig. 11A und 11B beschrieben wird. Nach dem Zustand F schreitet die ISS 100 zu dem Zustand D, um Instruktionsdecodieroperationen auszulösen. Im Zustand D gibt die ISS 100 ein Decodiersignal zu der Decodiersteuereinheit 112 aus. Während sie im Zustand D ist, liest die ISS 100 zusätzlich einen Operationscode aus, und zwar entsprechend einer decodierten Instruktion von dem Operationscodespeicherregistersatz 116. Basierend auf dem empfangenen Operationscode, schreitet die ISS 100 zu dem Zustand E oder zu dem Zustand M fort, um Instruktionsausführungsoperationen durchzuführen. Die ISS 100 schreitet zu dem Zustand E in dem Fall fort, daß die Instruktion in einem einzigen Taktzyklus ausgeführt werden kann; andernfalls schreitet die ISS 100 zu dem Zustand M für Multizyklusinstruktionsausführung fort. Bei den Instruktionsausführungsoperationen erzeugt die ISS 100 DOU-Steuersignale, AOU-Steuersignale und/oder Signale, die auf die Speicherzugriffslogik 102 gerichtet sind, um die Ausführung der Instruktionen zu erleichtern, die dem wiedergewonnenen bzw. ausgelesenen Operationscode entsprechen. Nachfolgend entweder den Zuständen E oder M, schreitet die ISS 100 zu dem Zustand W fort. Im Zustand W erzeugt die ISS 100 DOU-Steuersignale, AOU-Steuersignale und/oder Speicherschreibsignale, um ein Speichern eines Ergebnisses einer Instruktionsausführung zu erleichtern. Auf dem Zustand W wird deshalb als ein Schreib-zurück-Zustand Bezug genommen.

Fachleute werden erkennen, daß die Zustände F, D, E oder M und W einen vollständigen Instruktionsausführungszyklus aufweisen. Nach dem Zustand W schreitet die ISS 100 zu dem Zustand Y in dem Fall fort, daß ein Aussetzen einer Instruktionsausführung erforderlich ist. Der Zustand Y entspricht einem nichttägigen Zustand bzw. Ruhezustand, der z. B. in dem Fall erforderlich sein kann, daß eine T-Maschine 14 Zugriff zu dem S-Maschinensteicher 34 fordert bzw. benötigt. Nachfolgend zum Zustand Y oder nach dem Zustand W kehrt die ISS 100 in dem Fall, daß die Instruktionsausführung fortgesetzt werden soll, zu dem Zustand F zurück, um einen anderen Instruktionsausführungszyklus aufzunehmen.

Wie in Fig. 6 gezeigt, beinhaltet das Zustandsdiagramm ebenso einen Zustand I, der festgelegt ist, ein Interruptservicezustand zu sein. Bei der vorliegenden Erfindung empfängt die ISS 100 ein Interruptmeldungssignal von der Interruptlogik 106. Wie im folgenden unter Bezugnahme auf die Fig. 7 beschrieben werden wird, erzeugt die Interruptlogik 106 Übergangssteuersignale und speichert die Übergangssteuersignale in dem Prozeßsteuerwort innerhalb des Prozeßsteuerregistersatzes 122. Die Übergangssteuersignale weisen vorzugsweise darauf hin, welcher der Zustände F, D, E, M, W und Y unterbrechbar sind bzw. auf ein Interrupt reagieren, und auf ein Niveau bzw. eine Ebene einer Interruptpräzision, die in bzw. bei jedem unterbrechbaren Zustand erforderlich ist, und für jeden unterbrechbaren Zustand auf einen nächster Zustand, bei dem die Instruktionsausführung fortgesetzt werden soll und der dem Zustand I folgt. Falls die ISS 100 ein Interruptmeldungssignal empfängt, während sie in einem gegebenen Zustand ist, schreitet die ISS 100 zu dem Zustand I, falls das Übergangssteuersignal anzeigt, daß der gegenwärtige Zustand unterbrechbar ist bzw. auf ein Interrupt ansprechen kann. Andernfalls schreitet die ISS 100 fort als ob sie kein Interruptsignal empfangen hätte, bis sie einen unterbrechbaren Zustand erreicht.

Wenn einmal die ISS 100 zum Zustand I fortgeschritten ist, greift die ISS 100 vorzugsweise auf den Prozeßsteuerregistersatz 122 zu, um ein Interruptmaskierflag bzw. eine Interruptmaskiermarke zu setzen, und sie empfängt einen Interruptvektor. Nachdem der Interruptvektor wiedergewonnen bzw. ausgelesen ist, arbeitet die ISS 100 vorzugsweise über einen herkömmlichen Subroutinensprung zu einer Interrupthandhabungseinrichtung bzw. zu einem Interrupthandler, wie es durch den Interruptvektor beschrieben ist, den gegenwärtigen Interrupt ab.

Bei der vorliegenden Erfindung wird die Rekonfiguration der DRPU 32 in Antwort auf folgendes ausgelöst: 1) einen Rekonfigurationsinterrupt, der auf der externen Steuerleitung 48 aktiv ist; oder 2) die Ausführung einer Rekonfigurationsanweisung innerhalb einer Abfolge von Programminstruktionen. Bei der bevorzugten Ausführungsform führen sowohl der Rekonfigurationsinterrupt als auch die Ausführung einer Rekonfigurationsanweisung zu einem Subroutinensprung zu einer Rekonfigurationshandhabungseinrichtung bzw. zu einem Rekonfigurationshandler. Vorzugsweise speichert die Rekonfigurationshandhabungseinrichtung Programmzustandsinformationen und gibt eine Konfigurationsdatensatzadresse und das Rekonfigurationssignal zu der Rekonfigura-

tionslogik 104.

In dem Fall, daß es sich bei dem vorliegenden Interrupt nicht um einen Rekonfigurationsinterrupt handelt, schreitet die ISS 100 zu dem nächsten Zustand fort, wie dies durch die Übergangssteuersignale angezeigt wird, wenn der Interrupt empfangen worden ist, wodurch ein Instruktionsausführungszyklus wieder aufgenommen, vollendet oder ausgelöst wird.

Bei der bevorzugten Ausführungsform variiert der Satz an Zuständen, der durch die ISS 100 unterstützt wird, gemäß der Natur der ISA, für die die DRPU 32 konfiguriert ist. Somit würde der Zustand M für eine ISA nicht vorliegen, in der eine oder mehrere Instruktionen in einem einzigen Taktzyklus ausgeführt werden können, was der Fall wäre bei einer typischen Innenschleifen-ISA. Wie gezeigt, definiert das Zustandsdiagramm der Fig. 6 vorzugsweise die Zustände, die durch die ISS 100 festgelegt werden, um eine Allzweck-Außenschleifen-ISA zu realisieren. Zur Realisierung der Innenschleifen-ISA unterstützt die ISS 100 vorzugsweise mehrere Sätze von Zuständen F, D, E und W, und zwar parallel, wodurch eine pipelineartige Steuerung einer Instruktionsausführung in einer Art und Weise erleichtert wird, die leicht von Fachleuten verstanden wird. Bei der bevorzugten Ausführungsform ist die ISS 100 als eine CLB-basierte Zustandsmaschine realisiert, die die Zustände oder einen Untersatz von den Zuständen, die oben beschrieben wurden, in Übereinstimmung mit der gegenwärtig betrachteten ISA unterstützt.

Die Interruptlogik 106 umfaßt vorzugsweise eine Zustandsmaschine, die Übergangssteuersignale erzeugt und Interruptmeldeoperationen in Antwort auf ein Interruptsignal durchführt, das über die externe Steuerleitung 48 empfangen wird. Nimmt man Bezug auf Fig. 7, so ist ein Zustandsdiagramm gezeigt, das einen bevorzugten Satz von Zuständen zeigt, die durch die Interruptlogik 106 unterstützt werden. Die Interruptlogik 106 beginnt ihren Betrieb bzw. ihre Operation im Zustand P. Der Zustand P entspricht einer Einschalt-, Reset- oder Rekonfigurationsbedingung. In Antwort auf das Vollendungssignal, das durch die Rekonfigurationslogik 104 abgegeben wird, schreitet die Interruptlogik 106 zum Zustand A und liest das Interruptantwortsignal aus dem Architekturbeschreibungsspeicher 101. Die Interruptlogik 106 erzeugt dann das Übergangssteuersignal aus den Interruptantwortsignalen und speichert das Übergangssteuersignal in dem Prozeßsteuerregistersatz 122. Bei der bevorzugten Ausführungsform beinhaltet die Interruptlogik 106 ein CLB-basiertes programmierbares Logikfeld bzw. eine CLB-basierte "Programmable Logic Array" (PLA), um die Interruptantwortsignale zu empfangen und um die Übergangssteuersignale zu erzeugen. Nachfolgend zum Zustand A schreitet die Interruptlogik 106 zu dem Zustand B fort, um auf ein Interruptsignal zu warten. Nach dem Empfang eines Interruptsignals schreitet die Interruptlogik 106 zu dem Zustand C in dem Fall fort, daß das Interruptmaskierflag innerhalb des Prozeßsteuerregistersatzes 122 zurückgesetzt wird. Wenn einmal der Zustand C erreicht ist bzw. vorliegt, bestimmt die Interruptlogik 106 den Ursprung des Interrupts, eine Interruptpriorität und eine Adresse der Interrupthandhabungseinrichtung bzw. eine Interrupthandleradresse. In dem Fall, daß das Interruptsignal ein Rekonfigurationsinterrupt ist, schreitet die Interruptlogik 106 zu dem Zustand R und speichert eine Konfigurationsdatensatzadresse in dem Prozeßsteuerregistersatz 122. Nach dem Zustand R oder nachfolgend zu dem Zustand C schreitet die Interruptlogik 106, in dem Fall, daß das Interruptsignal nicht ein Rekonfigurationsinterrupt ist, zu dem Zustand N fort und speichert die Interrupthandleradresse bzw. die Adresse der Interrupthandhabungseinrichtung in dem Prozeßsteuerregistersatz 122. Die Interruptlogik 106 schreitet als nächstes zu dem Zustand X und gibt ein Interruptmeldeignal zu der ISS 100. Nachfolgend zu dem Zustand X kehrt die Interruptlogik 106 zum Zustand B zurück, um auf ein nächstes Interruptsignal zu warten.

Bei der bevorzugten Ausführungsform variiert der Pegel bzw. die Ebene der Interruptwartzeit bzw. der Interruptverzögerungszeit, wie sie durch die Interruptantwortsignale und folglich durch die Übergangssteuersignale spezifiziert ist, in Abhängigkeit von der gegenwärtigen ISA, für die die DRPU 32 konfiguriert worden ist. Zum Beispiel erfordert eine ISA, die auf Hochleistungs-Echtzeitbewegungssteuerung ausgerichtet ist, schnelle und vorhersehbare Interruptantwortfähigkeiten. Der Konfigurationsdatensatz, der einer derartigen ISA entspricht, beinhaltet deshalb vorzugsweise Interruptantwortsignale, die anzeigen, daß eine Unterbrechung mit geringer Wartezeit bzw. Verzögerungszeit erforderlich ist. Die entsprechenden Übergangssteuersignale identifizieren wiederum mehrere ISS-Zustände als unterbrechbar, wodurch es einem Interrupt ermöglicht wird, einen Instruktionsausführungszyklus auszusetzen bzw. aufzugeben, bevor der Instruktionsausführungszyklus vollendet ist. Im Gegensatz zu einer ISA, die auf Echtzeitbewegungssteuerung ausgerichtet ist, benötigt eine ISA, die auf Bildfaltungsoperationen ausgerichtet ist, Interruptantwortfähigkeiten, die gewährleisten, daß die Anzahl der Faltungsoperationen, die pro Zeiteinheit durchgeführt wird, maximiert ist. Der Konfigurationsdatensatz, der der Bildfaltungs-ISA entspricht, beinhaltet vorzugsweise Interruptantwortsignale, die spezifizieren, daß eine Unterbrechung mit langer Warte- bzw. Verzögerungszeit erforderlich ist. Die entsprechenden Übergangssteuersignale identifizieren einen Zustand W vorzugsweise als einen unterbrechbaren. In dem Fall, daß die ISS 100 mehrere Sätze von Zuständen F, D, E und W parallel unterstützt, identifizieren die Bildsteuersignale, wenn sie konfiguriert sind, um die Bildfaltungs-ISA zu realisieren, jeden Zustand W als unterbrechbar und spezifizieren weiter, daß das Abarbeiten bezüglich des Interrupts verzögert werden soll, bis jede der parallelen Instruktionsausführungszyklen ihre Zustand-W-Operationen beendet hat. Dies gewährleistet, daß eine ganze Gruppe von Instruktionen ausgeführt werden wird, bevor ein Interrupt abgearbeitet wird, wodurch vernünftige Fließband-Ausführungsleistungspegel bzw. gepipelte Ausführungsleistungspegel aufrechterhalten werden.

In einer Art und Weise, die analog zu dem Pegel der Interruptverzögerungszeit ist, variiert der Pegel bzw. das Niveau der Interruptpräzision, wie sie durch die Interruptantwortsignale spezifiziert werden, ebenso in Abhängigkeit von der ISA, für die die DRPU 32 konfiguriert ist. Zum Beispiel geben die Interruptantwortsignale in dem Fall, daß ein Zustand M als ein unterbrechbarer Zustand für eine Außenschleifen-ISA festgelegt ist, die unterbrechbare Multizyklusoperationen unterstützt, vorzugsweise vor, daß präzise Interrupts erforderlich sind. Die Übergangssteuersignale spezifizieren somit bzw. geben somit vor, daß Interrupts, die in dem Zustand M empfangen werden, als präzise Interrupts behandelt werden, um zu gewährleisten, daß Multizyklusoperationen

erfolgreich erneut gestartet werden können. In einem anderen Beispiel bezüglich einer ISA, die nicht fehlbare bzw. nicht fehlerbehaftete Fließband-Arithmetikoperationen unterstützt, spezifizieren Interruptantwortsignale vorzugsweise, daß ungenaue bzw. unpräzise Interrupts erforderlich sind. Die Übergangssteuersignale spezifizieren dann bzw. geben dann vor, daß die Interrupts, die im Zustand W empfangen werden, als unpräzise Interrupts behandelt werden.

Bezüglich jeder gegebenen ISA werden die Interruptantwortsignale durch einen Abschnitt des Datensatzes, der der ISA entspricht, festgelegt oder programmiert. Über programmierbare Interruptantwortsignale und die Erzeugung von entsprechenden Übergangssteuersignalen erleichtert die vorliegende Erfindung die Realisierung eines optimalen Interruptschemas auf einer ISA-durch-ISA-Basis. Fachleute werden erkennen, daß die überwiegende Mehrzahl von Computerarchitekturen nach dem Stand der Technik nicht für die flexible Spezifizierung von Unterbrechungsfähigkeiten bzw. Interruptfähigkeiten, nämlich programmierbare Zustands-Übergangsfreigabe, programmierbare Interruptverzögerungszeit und programmierbare Interruptpräzision sorgen. Bei der bevorzugten Ausführungsform ist die Interruptlogik 106 als eine CLB-basierte Zustandsmaschine realisiert, die die oben beschriebenen Zustände unterstützt.

Die Abrufsteuereinheit 108 verwaltet das Laden von Instruktionen in den Instruktionspuffer 110 in Antwort auf das Abrufsignal, das von der ISS 100 ausgegeben wird. Bei der bevorzugten Ausführungsform ist die Abrufsteuereinheit 108 als eine herkömmliche "One-Hot" codierte bzw. Monoflog-kodierte Zustandsmaschine realisiert, die Flipflops innerhalb eines Satzes von CLBs verwendet. Fachleute werden erkennen, daß bei einer alternativen Ausführungsform die Abrufsteuereinheit 108 als eine konventionell codierte Zustandsmaschine oder als eine ROM-basierte Zustandsmaschine konfiguriert werden könnte. Der Instruktionspuffer 110 stellt eine temporäre Speicherung für Instruktionen bereit, die von dem Speicher 34 geladen werden. Für die Realisierung einer Außenschleifen-ISA ist der Instruktionspuffer 110 vorzugsweise als herkömmlicher RAM-basierter "Zuerst-rein, Zuerst-raus"- bzw. "First In, First Out"- (FIFO)-Puffer realisiert, der eine Vielzahl von CLBs verwendet. Für die Realisierung einer Innenschleifen-ISA ist der Instruktionspuffer 110 vorzugsweise als ein Satz von Flipflop-Registern realisiert, die eine Vielzahl von Flipflops innerhalb eines Satzes von IOBs oder eine Vielzahl von Flipflops innerhalb IBOs als auch CLBs verwendet.

Die Decodersteuereinheit 112 verwaltet den Transfer von Instruktionen von dem Instruktionspuffer 110 in den Instruktionsdecoder 114 in Antwort auf das Decodiersignal, das von der ISS 100 ausgegeben wird. Bezüglich einer Innenschleifen-ISA ist die Decodiersteuereinheit 112 vorzugsweise als eine Zustandsmaschine auf ROM-Basis realisiert, die ein ROM auf CLB-Basis aufweist, das mit einem Register auf CLB-Basis verbunden ist. Bezüglich einer Außenschleifen-ISA ist die Decodiersteuereinheit 112 vorzugsweise als eine codierte Zustandsmaschine auf CLB-Basis realisiert. Bezüglich jeder Instruktion, die als Eingang empfangen wird, gibt der Instruktionsdecoder 114 einen entsprechenden Operationscode bzw. Opcode, eine Registerfileadresse und optional eine oder mehrere Konstanten in einer herkömmlichen Art und Weise aus. Bezüglich einer Innenschleifen-ISA ist der Instruktionsdecoder 114 vorzugsweise konfiguriert, um eine Gruppe von Instruktionen zu decodieren, die als ein Eingang bzw. ein Eingangssignal empfangen werden. Bei der bevorzugten Ausführungsform ist der Instruktionsdecoder 114 als ein Decoder auf CLB-Basis bzw. als ein CLB-basierter Decoder konfiguriert, um jede der Instruktionen zu decodieren, die in der ISA, die gegenwärtig betrachtet wird, beinhaltet sind.

Der Operationscode-Speicherregistersatz 116 stellt eine temporäre Speicherung für jeden Operationscode bereit, der durch den Instruktionsdecoder 114 ausgegeben wird und gibt jeden Operationscode bzw. Opcode zu der ISS 100 aus. Wenn eine Außenschleifen-ISA in der DRPU 32 realisiert wird, wird der Operationscode-Speicherregistersatz 116 vorzugsweise realisiert, indem eine optimale Anzahl an Flipflop-Registerbänken verwendet wird. Die Flipflop-Registerbänke empfangen Signale von dem Instruktionsdecoder 114, die Klassen- oder Gruppencodes darstellen, die von Operationscode-Literal-Bitfeldern von Instruktionen abgeleitet werden, die zuvor durch den Instruktionspuffer 110 geschleust wurden bzw. dort in einer Warteschlange eingereiht wurden. Die Flipflop-Registerbänke speichern die zuvor erwähnten Klassen- oder Gruppencodes gemäß einem Decodierschema, das vorzugsweise die ISS-Komplexität minimiert. Für den Fall einer Innenschleifen-ISA speichert der Operationscode-Speicherregistersatz 116 vorzugsweise Operationscode-Hinweissignale, die direkter von den Operationscode-Bitfeldern abgeleitet werden, die durch den Instruktionsdecoder 114 ausgegeben werden. Innenschleifen-ISAs weisen notwendigerweise kleinere Operationscode-Literal-Bitfelder auf, wodurch die Realisierungserfordernisse für das Puffern, Decodieren bzw. Operationscode-Anzeigen für das Sequenzieren von Instruktionen durch den Instruktionspuffer 110, den Instruktionsdecodierer 114 bzw. den Operationscode-Speicherregistersatz 116 minimiert werden. Zusammengefaßt ist bezüglich Außenschleifen-ISAs der Operationscode-Speicherregistersatz 116 vorzugsweise als ein kleiner Verbund von Flipflop-Registerbänken realisiert, die durch eine Bitbreite charakterisiert sind, die gleich der Operationscode-Literal-Größe ist oder einen Bruchteil davon darstellt. Bezüglich Innenschleifen-ISAs ist der Operationscode-Speicherregistersatz 116 vorzugsweise eine kleinere und gleichmäßige Flipflop-Registerbank als bezüglich Außenschleifen-ISAs. Die reduzierte Flipflop-Registerbankgröße in dem Innenschleifenfall spiegelt die minimale Instruktionszählcharakteristik von Innenschleifen-ISAs relativ zu Außenschleifen-ISAs wieder.

Der RF-AdresseRegistersatz 118 bzw. der KonstantenRegistersatz 120 stellen eine temporäre Speicherung für jede Registerfileadresse bzw. für jede Konstante bereit, die durch den Instruktionsdecoder 114 ausgegeben wird. Bei der bevorzugten Ausführungsform werden der Operationscode-Speicherregistersatz 116, der RF-AdresseRegistersatz 118 und der KonstantenRegistersatz 120 jeweils als ein Satz von CLBs realisiert, die für die Datenspeicherung konfiguriert sind.

Bei der Speicherzugriffslogik 102 handelt es sich um eine Speicher-Steuerschaltung, die den Transfer bzw. die Übertragung von Daten zwischen dem Speicher 34, der DOU 62 und der AOU 64 gemäß der atomaren Speicheradressengröße verwaltet und synchronisiert, die in dem Architekturbeschreibungsspeicher 122 spezifi-

ziert ist. Die Speicherzugriffslogik 102 verwaltet und synchronisiert zusätzlich den Transfer von Daten und Befehlen bzw. Kommandos zwischen der S-Maschine 12 und einer gegebenen T-Maschine 14. Bei der bevorzugten Ausführungsform unterstützt die Speicherzugriffslogik 102 Burstdodus-Speicherzugriffe und sie ist vorzugsweise als eine herkömmliche RAM-Steuereinrichtung realisiert, die CLBs verwendet. Fachleute werden erkennen, daß während der Rekonfiguration die Eingangs- und Ausgangspins bzw. -anschlüsse der Rekonfigurierbaren Logikvorrichtung drei Zustände aufweisen, die ohmsche Abschlüsse ermöglichen, um unaktive bzw. nicht angesteuerte Logikpegel festzulegen, und sie werden somit nicht den Speicher 34 stören. Bei einer alternativen Ausführungsform könnte die Speicherzugriffslogik 102 außerhalb der DRPU 32 realisiert sein.

Nimmt man nun Bezug auf Fig. 8, so ist eine Datenoperationseinheit bzw. "Data Operate Unit" 62 gezeigt. Die DOU 62 führt bezüglich der Daten Operationen gemäß den DOU-Steuersignalen, RF-Adressen und Konstanten durch, die von der ISS 100 empfangen werden. Die DOU 62 umfaßt einen DOU-Crossbar-Schalter (X-Bar-Schalter) 150, eine Abspeicher-/Ausrichtlogik 152 und eine Datenoperationslogik 154. Sowohl der DOU-Crossbar-Schalter 150 als auch die Abspeicher-/Ausrichtlogik 152 und die Datenoperationslogik 154 umfassen einen Steuereingang, der mit dem ersten Steuerausgang der IFU 60 über die erste Steuerleitung 70 verbunden ist. Der DOU-Crossbar-Schalter 150 umfaßt einen bidirektionalen Datenport, der den bidirektionalen Datenport des DOU ausbildet; einen Konstanteneingang, der mit der dritten Steuerleitung 74 verbunden ist; einen ersten Datenrückführeingang bzw. Daten-Feedback-Eingang, der mit einem Datenausgang der Datenoperationslogik 154 über eine erste Datenleitung 160 verbunden ist; einen zweiten Datenrückführeingang bzw. Daten-Feedback-Eingang, der mit einem Datenausgang der Abspeicher-/Ausrichtlogik 152 über eine zweite Datenleitung 164 verbunden ist; und einen Datenausgang, der mit einem Dateneingang der Abspeicher-/Ausrichtlogik 152 über eine dritte Datenleitung 162 verbunden ist. Zusätzlich zu ihren Datenausgängen umfaßt die Abspeicher-/Ausrichtlogik 154 einen Adresseneingang, der mit der dritten Steuerleitung 74 verbunden ist. Die Datenoperationslogik 154 weist zusätzlich einen Dateneingang auf, der mit dem Ausgang der Abspeicher-/Ausrichtlogik über die zweite Datenleitung 164 verbunden ist.

Die Datenoperationslogik 154 führt arithmetische, Schiebe- und/oder logische Operationen bezüglich der Daten durch, die bei ihrem Dateneingang in Antwort auf die DOU-Steuersignale empfangen werden, die bei ihrem Steuereingang empfangen werden. Die Abspeicher-/Ausrichtlogik 152 umfaßt Datenspeicherelemente, die eine temporäre Speicherung bezüglich Operanden, Konstanten und Teilergebnissen, die mit Datenberechnungen verbunden sind, bereitstellen, und zwar unter der Verwaltung der RF-Adressen bzw. DOU-Steuersignale, die bei ihrem Adresseneingang bzw. Steuereingang empfangen werden. Der DOU-Crossbar-Schalter 150 ist vorzugsweise ein konventionelles Crossbar-Schalt-Netzwerk, das das Laden von Daten aus dem Speicher 34, den Transfer von Ergebnissen, die durch die Datenoperationslogik 154 ausgegeben werden, zu der Abspeicher-/Ausrichtlogik 152 des Speichers 34, und das Laden von Konstanten, die durch die IFU 60 ausgegeben werden, in die Abspeicher-/Ausrichtlogik 152 in Übereinstimmung mit DOU-Steuersignalen, die bei ihrem Steuereingang empfangen werden, erleichtert. Bei der bevorzugten Ausführungsform hängt die detaillierte Struktur der Datenoperationslogik 154 von den Typen der Operationen ab, die durch die ISA, die gegenwärtig betrachtet wird, unterstützt werden. Das heißt, die Datenoperationslogik 154 umfaßt eine Schaltung, um arithmetische und/oder logische Operationen durchzuführen, die durch die Datenoperationsinstruktionen innerhalb der gegenwärtig betrachteten ISA spezifiziert werden. In ähnlicher Weise hängt die detaillierte Struktur der Abspeicher-/Ausrichtlogik 152 und des DOU-Crossbar-Schalters 150 von der gegenwärtig betrachteten ISA ab. Die detaillierte Struktur der Datenoperationslogik 154, der Abspeicher-/Ausrichtlogik 152 und der DOU-Crossbar-Schalters 150 gemäß des ISA-Typs wird im folgenden unter Bezugnahme auf die Fig. 9A und 9B beschrieben.

Bezüglich einer Außenschleifen-ISA ist die DOU 62 vorzugsweise konfiguriert, um serielle Operationen bezüglich der Daten durchzuführen. Nimmt man Bezug auf die Fig. 9A, so ist ein Blockdiagramm bzw. ein Blockschaltbild einer ersten beispielhaften Ausführungsform der DOU 61 gezeigt, die für die Realisierung einer Allzweck-Außenschleifen-ISA konfiguriert ist. Eine Allzwecke-Außenschleifen-ISA erfordert eine Hardware, die konfiguriert ist, um mathematische Operationen, wie z. B. Multiplikation, Addition und Subtraktion; boolsche Operationen, wie z. B. AND bzw. UND, ODER bzw. OR und NICHT bzw. NOT; Schiebeoperationen; und Rotationsoperationen bzw. Drehoperationen durchzuführen. Somit weist die Datenoperationslogik 154 für die Realisierung einer Allzweck-Außenschleifen-ISA vorzugsweise eine herkömmliche arithmetische Logikeinheit bzw. "Arithmetic-Logic Unit" (ALU)/Verschiebeeinrichtung 184 mit einem ersten Eingang, einem zweiten Eingang, einem Steuereingang und einem Ausgang auf. Die Abspeicher-/Ausrichtlogik 152 umfaßt vorzugsweise einen ersten RAM 180 und einen zweiten RAM 182 von denen jeder einen Dateneingang, einen Datenausgang, einen Adress-Wähleingang und einen Freigabeeingang bzw. Enable-Eingang aufweist. Der DOU-Crossbar-Schalter 150 umfaßt vorzugsweise ein herkömmliche Crossbar-Schaltnetzwerk mit zwei bidirektionalen und unidirektionalen Crossbar-Verbindungen und mit Eingängen und Ausgängen, die zuvor unter Bezugnahme auf die Fig. 8 beschrieben wurden. Fachleute werden erkennen, daß eine wirksame Realisierung des DOU-Crossbar-Schalters 150 für eine Außenschleifen-ISA, Multiplexer, Drei-Zustands-Puffer, CLB-basierte Logik, direkte Verdrahtung oder Untersätze bzw. Subsets von zuvor erwähnten Elementen beinhaltet, die in jeglicher Kombination durch die Wirkung bzw. Eigenschaft der rekonfigurierbaren Kopplungseinrichtung bzw. rekonfigurierbaren Verbindungseinrichtung verbunden werden. Bezüglich einer Außenschleifen-ISA wird der DOU-Crossbar-Schalter 150 realisiert bzw. implementiert, um eine serielle Datenbewegung in einer minimal möglichen Zeit voranzutreiben bzw. zu beschleunigen, während ebenso eine maximale Anzahl von einzigen Datenbewegungs-Crossbar-Verbindungen bereitgestellt werden, um verallgemeinerte Außenschleifen-Instruktionstypen zu unterstützen.

Der Dateneingang des ersten RAM's 180 ist ebenso wie der Dateneingang des zweiten RAM's 182 über die dritte Datenleitung 162 mit dem Datenausgang des DOU-Crossbar-Schalters 150 verbunden. Die Adressenauswahleingänge des ersten RAM's 180 und des zweiten RAM's 182 sind verbunden, um Registerfileadressen von

der IFU 60 über die dritte Steuerleitung 74 zu empfangen. In ähnlicher Weise sind die Freigabeeingänge des ersten und des zweiten RAM's 180, 182 verbunden, um DOU-Steuersignale über die erste Steuerleitung 70 zu empfangen. Die Datenausgänge des ersten bzw. des zweiten RAM's 180, 182 sind mit dem ersten bzw. dem zweiten Eingang der ALU/Verschiebeeinrichtung 184 verbunden und sind ebenso mit dem zweiten Daten-Feedback-Eingang des DOU-Crossbar-Schalters 150 verbunden. Der Steuereingang der ALU/Verschiebeeinrichtung 184 ist verbunden, um die DOU-Steuersignale über die erste Steuerleitung 70 zu empfangen, und der Ausgang der ALU/Verschiebeeinrichtung 184 ist mit dem ersten Datenrückführriegang des DOU-Crossbar-Schalters 150 verbunden. Die Verbindungen zu den übrigen Eingängen und Ausgängen des DOU-Crossbar-Schalters 150 sind mit jenen identisch, die oben unter Bezugnahme auf die Fig. 8 beschrieben wurden.

Um die Ausführung einer Datenoperationsinstruktion zu erleichtern, gibt die IFU 60 DOU-Steuersignale, RF-Adressen und Konstante zu der DOU 61 während einem der beiden ISS-Zustände E oder M aus. Der erste bzw. zweite RAM 180, 182 liefern ein erstes bzw. zweites Registerfile zur temporären Datenspeicherung. Individuelle Adressen innerhalb des ersten und des zweiten RAM's 180, 182 werden gemäß den RF-Adressen ausgewählt, die bei dem jeweiligen Adresseneingang jedes RAM's empfangen werden. In ähnlicher Weise wird das Laden des ersten und zweiten RAM's 180, 182 durch die DOU-Steuersignale gesteuert, die jedes RAM 180, 182 bei seinem Schreibfreigabeeingang empfängt. Bei der bevorzugten Ausführungsform beinhaltet wenigstens ein RAM 180, 182 eine Hindurchreichfähigkeit, um den Transfer von Daten von dem DOU-Crossbar-Schalter 150 direkt zu der ALU/Verschiebeeinrichtung 184 zu erleichtern. Die ALU/Verschiebeeinrichtung 184 führt arithmetische, logische oder Schiebeoperationen bezügliche eines ersten Operanden aus, der von dem ersten RAM 180 empfangen wird, und/oder bezüglich eines zweiten Operanden aus, der von dem zweiten RAM 182 empfangen wird, und zwar unter der Verwaltung der DOU-Steuersignale, die bei ihrem Steuereingang empfangen werden. Der DOU-Crossbar-Schalter 150 leitet selektiv folgendes: 1) Daten zwischen dem Speicher 34 und dem ersten und zweiten RAM 180, 182; 2) Ergebnisse von der ALU/Verschiebeeinrichtung 184 zu dem ersten und zweiten RAM 180, 182 oder dem Speicher 34; 3) Daten, die in dem ersten oder zweiten RAM 180, 182 gespeichert sind, zu dem Speicher 34; und 4) Konstanten von der IFU 60 zu dem ersten und zweiten RAM 180, 182. Wie zuvor beschrieben wurde, leitet in dem Fall, daß entweder der erste oder der zweite RAM 180, 182 eine Hindurchreichfähigkeit beinhaltet, der DOU-Crossbar-Schalter 150 ebenso selektiv Daten von dem Speicher 34 oder dem Ausgang der ALU/Verschiebeeinrichtung direkt zurück in die ALU/Verschiebeeinrichtung 184. Der DOU-Crossbar-Schalter 150 führt eine bestimmte Leitweg-Operation bzw. Wegeermittlungsoperation gemäß den DOU-Steuersignalen durch, die bei seinem Steuereingang empfangen werden. Bei der bevorzugten Ausführungsform wird die ALU/Verschiebeeinrichtung 184 realisiert, indem Logikfunktionsgeneratoren innerhalb eines Satzes von CLBs und eine Schaltung, die auf mathematische Operationen innerhalb der rekonfigurierbaren Logikvorrichtung ausgerichtet ist, verwendet werden. Der erste und zweite RAM 180, 182 werden vorzugsweise realisiert, indem die Datenspeicherschaltung, die in einem Satz von CLBs vorliegt, verwendet wird, und der DOU-Crossbar-Schalter 150 wird vorzugsweise in einer Art und Weise realisiert, wie vorstehend beschrieben.

Es wird nun auf die Fig. 9B Bezug genommen. Dort ist ein Blockdiagramm bzw. ein Blockschaltbild einer zweiten beispielhaften Ausführungsform der DOU 63 gezeigt, die für die Realisierung einer Innenschleifen-ISA konfiguriert ist. Im allgemeinen unterstützt eine Innenschleifen-ISA relativ wenige spezialisierte Operationen und wird vorzugsweise verwendet, um einen allgemeinen Satz von Operationen bezüglich potentiell großer Datensätze durchzuführen. Optimale Rechenleistungsfähigkeit wird deshalb für eine Innenschleifen-ISA durch eine Hardware erzeugt, die konfiguriert ist, um Operationen parallel durchzuführen. Somit sind bei der zweiten beispielhaften Ausführung der DOU 63 die Datenoperationslogik 154, die Abspeicher-/Ausrichtlogik 152 und der DOU-Crossbar-Schalter 150 konfiguriert, um Fließband-Berechnungen bzw. gepipelinetre Berechnungen durchzuführen. Die Datenoperationslogik 154 umfaßt eine Fließband-Funktionseinheit bzw. gepipelinetre Funktionseinheit 194 mit einer Anzahl von Eingängen, einem Steuereingang und einem -ausgang. Die Abspeicher-/Ausrichtlogik 152 umfaßt folgendes: 1) einen Satz von herkömmlichen Flipflop-Feldern 192, wobei jedes Flipflop-Feld 192 einen Dateneingang, einen Datenausgang und einen Steuereingang umfaßt; und 2) eine Datenauswähleinrichtung 190 mit einem Steuereingang, einem Dateneingang und einer Anzahl von Datenausgängen, die der Anzahl der vorliegenden Flipflop-Felder 192 entsprechen. Der DOU-Crossbar-Schalter 150 umfaßt ein herkömmliches Crossbar-Schaltwerk mit Duplex-unidirektionalem Crossbar-Verbindungen. Bei der zweiten beispielhaften Ausführungsform der DOU 63 beinhaltet der DOU-Crossbar-Schalter 150 vorzugsweise die Eingänge und Ausgänge, die vorhergehend unter Bezugnahme auf die Fig. 8 beschrieben wurden, und zwar mit Ausnahme des zweiten Datenrückführriegangs. In einer Art und Weise, die dem Außenschleifen-ISA-Fall analog ist, kann eine wirksame Realisierung des DOU-Crossbar-Schalters 150 für eine Innenschleifen-ISA, Multiplexer, Drei-Zustands-Puffer, eine CLB-basierte Logik, eine direkte Verdrahtung bzw. eine direkte Leitung oder einen Untersatz der zuvor erwähnten Elemente, die in rekonfigurierbarer Art und Weise verbunden sind, beinhalten. Für eine Innenschleifen-ISA ist der DOU-Crossbar-Schalter 150 vorzugsweise realisiert, um eine parallele Datenbewegung innerhalb eines minimalen Zeitaufwands zu maximieren, während ebenso eine minimale Anzahl von einzigen Datenbewegungs-Crossbar-Verbindungen bereitgestellt wird, um Innenschleifen-ISA-Instruktionen zu unterstützen, die eine starke Betonung der Fließbandbearbeitung aufweisen bzw. die stark "gepipelined" sind.

Der Dateneingang der Datenauswähleinrichtung 190 ist mit dem Datenausgang des DOU-Crossbar-Schalters 150 über die erste Datenleitung 162 verbunden. Der Steuereingang der Datenauswähleinrichtung 190 ist verbunden, um RF-Adressen über die dritte Steuerleitung 74 zu empfangen, und jeder Ausgang der Datenauswähleinrichtung 190 ist mit einem entsprechenden Flipflop-Feld-Dateneingang verbunden. Der Steuereingang eines jeden Flipflop-Feldes 192 ist verbunden, um DOU-Steuersignale über die erste Steuerleitung 70 zu empfangen, und jeder Flipflop-Feld-Datenausgang ist mit einem Eingang der Funktionseinheit 194 verbunden. Der Steuereingang der Funktionseinheit 194 ist verbunden, um DOU-Steuersignale über die erste Steuerleitung 70 zu

empfangen, und der Ausgang der Funktionseinheit 194 ist mit dem ersten Datenrückführeingang des DOU-Crossbar-Schalters 150 verbunden. Die Verbindungen der verbleibenden Eingänge und Ausgänge des DOU-Crossbar-Schalters 150 sind mit jenen identisch, die zuvor unter Bezugnahme auf die Fig. 8 beschrieben wurden.

Im Betrieb führt die Funktionseinheit 194 Fließband-Operationen bzw. gepipelnete Operationen bezüglich der Daten durch, die bei ihren Dateneingängen in Übereinstimmung mit den DOU-Steuersignalen empfangen werden, die bei ihrem Steuereingang empfangen werden. Fachleute werden erkennen, daß die Funktionseinheit 194 eine Multiplikations-Akkumulationseinheit, eine Schwellen-Bestimmungseinheit, eine Bild-Rotationseinheit, eine Kantenverstärkungseinheit bzw. eine Randverstärkungseinheit oder irgendeine Art von Funktionseinheit sein kann, die geeignet ist, um Fließband-Operationen bzw. gepipelnete Operationen bezüglich der aufgeteilten bzw. partitionierten Daten durchzuführen. Die Datenauswähleinrichtung 190 leitet Daten von dem Eingang des DOU-Crossbar-Schalters 150 in ein gegebenes Flipflop-Feld 192 gemäß den RF-Adressen, die bei seinem Steuereingang empfangen werden. Jedes Flipflop-Feld 192 beinhaltet vorzugsweise einen Satz von sequentiell gekoppelten Datenlatches bzw. Datenhalteeinrichtungen, um räumlich und temporär Daten relativ zu dem Dateninhalt eines anderen Flipflop-Feldes 192 auszurichten, und zwar unter der Verwaltung der Steuersignale, die bei seinem Steuereingang empfangen werden. Der DOU-Crossbar-Schalter 150 leitet selektiv folgendes: 1) Daten von dem Speicher 34 zu der Datenauswähleinrichtung 190; 2) Ergebnisse von der Multiplikations-/Akkumulationseinheit 194 zu der Datenauswähleinrichtung 190 oder dem Speicher 34; und 3) Konstante von der IFU 60 zu der Datenauswähleinrichtung 190. Fachleute werden erkennen, daß eine Innenschleifen-ISA einen Satz von eingebauten bzw. "built-in"-Konstanten aufweist. Bei der Realisierung einer derartigen Innenschleifen-ISA beinhaltet die Abspeicher-/Ausrichtlogik 154 vorzugsweise ein ROM auf CLB-Basis bzw. ein CLB-basiertes ROM, das eingebaute Konstanten enthält, wodurch das Erfordernis beseitigt wird, die Konstanten von der IFU 60 in die Abspeicher-/Ausrichtlogik 152 über den DOU-Crossbar-Schalter 150 zu leiten. Bei der bevorzugten Ausführungsform ist die Funktionseinheit 194 vorzugsweise implementiert bzw. realisiert, indem Logikfunktionsgeneratoren und eine Schaltung, die auf mathematische Operationen innerhalb eines Satzes von CLBs ausgerichtet ist, verwendet werden. Jedes Flipflop-Feld 192 ist vorzugsweise realisiert, indem Flipflops innerhalb eines Satzes von CLBs verwendet werden, und die Datenauswähleinrichtung 190 ist vorzugsweise implementiert, indem Logikfunktionsgeneratoren und eine Datenauswählshaltung innerhalb eines Satzes von CLBs verwendet werden. Schließlich wird die Codier- und Decodierschaltung 150 vorzugsweise in der zuvor für eine Innenschleifen-ISA beschriebenen Art und Weise realisiert.

Nimmt man nun Bezug auf Fig. 10, so ist ein Blockdiagramm einer bevorzugten Ausführungsform der Adressenoperationseinheit 64 gezeigt. Die AOU 64 führt Operationen bezüglich der Adressen gemäß der AOU-Steuersignale, RF-Adressen und Konstanten durch, die von der IFU 60 empfangen werden. Die AOU 64 umfaßt einen AOU-Crossbar-Schalter 200, eine Abspeicher-/Zähllogik 202, eine Adressenoperationslogik 204 und einen Adressen-Multiplexer 206. Sowohl der AOU-Crossbar-Schalter 200 als auch die Abspeicher-/Zähllogik 202 und die Adressenoperationslogik 204 und der Adressenmultiplexer 206 umfassen einen Steuereingang, der mit dem zweiten Steuereingang des IFU 60 über die zweiten Steuerleitung 72 verbunden ist. Der AOU-Crossbar-Schalter 200 umfaßt einen bidirektionalen Datenport, der den bidirektionalen Datenport des AOU ausbildet; einen Adressen-Rückführeingang, der mit einem Adressenausgang der Adressenoperationslogik 204 über eine erste Adressenleitung 201 verbunden ist; einen Konstanteneingang, der mit der dritten Steuerleitung 74 verbunden; und einen Adressenausgang, der mit einem Adresseneingang über der Abspeicher-/Zähllogik 202 über eine zweite Adressenleitung 212 verbunden ist. Zusätzlich zu seinem Adresseneingang und Steuereingang, weist die Abspeicher-/Zähllogik 202 einen RF-Adresseneingang auf, der mit der dritten Steuerleitung 74 verbunden ist, und einen Adressenausgang auf, der mit einem Adresseneingang der Adressenoperationslogik 204 über eine dritte Adressenleitung 214 verbunden ist. Der Adressenmultiplexer 206 umfaßt einen ersten Eingang, der mit der ersten Adressenleitung 210 verbunden ist, einen zweiten Eingang, der mit der dritten Adressenleitung 214 verbunden ist, und einen Ausgang, der den Adressenausgang der AOU 64 ausbildet.

Die Adressenoperationslogik 204 führt arithmetische Operationen bezüglich der Adressen durch, die bei ihrem Adresseneingang empfangen werden, und zwar unter der Verwaltung der AOU-Steuersignale, die bei ihrem Steuereingang empfangen werden. Die Abspeicher-/Zähllogik 202 stellt eine temporäre Speicherung von Adressen bereit und adressiert Rechenergebnisse. Der AOU-Crossbar-Schalter 200 erleichtert das Laden von Adressen von dem Speicher 34, den Transfer von Ergebnissen, die durch die Adressenoperationslogik 204 ausgegeben werden, zu der Abspeicher-/Zähllogik 202 oder dem Speicher 34, und das Laden von Konstanten, die durch die IFU 60 ausgegeben werden, in die Abspeicher-/Zähllogik 202 in Übereinstimmung mit den AOU-Steuersignalen, die bei ihrem Eingang empfangen werden. Der Adressenmultiplexer 206 gibt selektiv eine Adresse, die von der Abspeicher-/Zähllogik 202 oder der Adressenoperationslogik 204 empfangen wird, zu dem Adressenausgang der AOU 64 unter der Verwaltung des AOU-Steuersignals aus, das bei seinem Steuereingang empfangen wird. Bei der bevorzugten Ausführungsform hängt die detaillierte Struktur des AOU-Crossbar-Schalters 200, der Abspeicher-/Ausrichtlogik 202 und der Adressenoperationseinheit 204 von dem Typ der ISA ab, die gegenwärtig betrachtet wird, wie im folgenden unter Bezugnahme auf die Fig. 11A und 11B beschrieben wird.

Es wird nun auf die Fig. 11A Bezug genommen. Dort ist ein Blockschaltbild einer ersten beispielhaften Ausführungsform der AOU 65 gezeigt, die für die Realisierung einer Allzweck-Außenschleifen-ISA konfiguriert ist. Eine Allzweck-Außenschleifen-ISA erfordert eine Hardware zur Durchführung von Operationen, wie z. B. Addition, Subtraktion, Inkrementation und Dekrementation bezüglich der Inhalte eines Programmzählers und bezüglich Adressen, die in der Abspeicher-/Zähllogik 202 gespeichert sind. Bei der ersten beispielhaften Ausführungsform der AOU 65 umfaßt die Adressenoperationslogik 204 vorzugsweise ein Programmadressenregister bezüglich nächster Befehle bzw. ein "Next Instruction Program Address Register" (NIPAR) 232 mit einem Eingang, einem Ausgang und einem Steuereingang; wobei eine arithmetische Einheit 234 einen ersten Eingang,

einen zweiten Eingang, einen dritten Eingang, einen Steuereingang und einen Ausgang umfaßt; und ein Multiplexer 230 einen ersten Eingang, einen zweiten Eingang, einen Steuereingang und einen Ausgang umfaßt. Die Abspeicher-/Zähllogik 202 umfaßt vorzugsweise einen dritten RAM 220 und einen vierten RAM 222, wobei jeder einen Eingang, einen Ausgang und einen Adressenauswahleingang und einen Freigabeeingang umfaßt.

5 Der Adressenmultiplexer 206 umfaßt vorzugsweise einen Multiplexer mit einem ersten Eingang, einem zweiten Eingang, einem dritten Eingang, einem Steuereingang und einem Ausgang. Der AOU-Crossbar-Schalter 200 umfaßt vorzugsweise ein herkömmliches Crossbar-Schaltnetzwerk mit Duplex-unidirektionalen Crossbar-Verbindungen und mit den Eingängen und Ausgängen, die zuvor unter Bezugnahme auf die Fig. 10 beschrieben wurden. Eine wirksame Realisierung des AOU-Crossbar-Schalters 200 kann Multiplexer, Drei-Zustandspuffer, 10 eine CLB-basierte Logik, eine Direktverdrahtung bzw. eine Direktleitung oder irgendeinen Untersatz von derartigen Elementen aufweisen, die durch rekonfigurierbare Verbindungen verbunden sind. Für eine Außenschleifen-ISA wird der AOU-Crossbar-Schalter vorzugsweise realisiert, um die Bewegung bezüglich serieller Adressen zu maximieren, und zwar innerhalb eines minimalen Zeitaufwands, während ebenso eine maximale Anzahl von einzigen bzw. eindeutigen Adressenbewegungs-Crossbar-Verbindungen bereitgestellt wird, um 15 verallgemeinerte Außenschleifen-ISA-Adressenoperationsinstruktionen zu unterstützen.

Der Eingang des dritten RAM's 220 und der Eingang des vierten RAM's 222 sind jeweils mit dem Ausgang des AOU-Crossbar-Schalters 200 über die zweite Adressenleitung 212 verbunden. Die Adressenauswahleingänge des dritten und vierten RAM's 220, 222 sind verbunden, um RF-Adressen von der IFU 60 über die dritte Steuerleitung 74 zu empfangen, und die Freigabeeingänge des ersten und zweiten RAM's 220, 222 sind verbunden, um AOU-Steuersignale über die zweite Steuerleitung 72 zu empfangen. Der Ausgang des dritten RAM's 220 ist mit dem ersten Eingang des Multiplexers 230, dem ersten Eingang der arithmetischen Einheit 234 und dem ersten Eingang des Adressenmultiplexers 206 verbunden. In ähnlicher Weise ist der Ausgang des vierten RAM's 222 mit dem zweiten Eingang des Multiplexers 230, dem zweiten Eingang der arithmetischen Einheit 234 und dem zweiten Eingang des Adressenmultiplexers 206 verbunden. Die Steuereingänge des Multiplexers 230, 20 25 der NIPAR 232 und der arithmetischen Einheit 234 sind jeweils mit der zweiten Steuerleitung 72 verbunden. Der Ausgang der arithmetischen Einheit 234 bildet den Ausgang der Adressenoperationslogik 204 und ist deshalb mit dem Adressen-Rückföhreingang des AOU-Crossbar-Schalters 200 und dem dritten Eingang des Adressenmultiplexers 206 verbunden. Die Verbindungen zu den übrigen Eingängen und Ausgängen des AOU-Crossbar-Schalters 200 und des Adressenmultiplexers 206 sind mit jenen identisch, die zuvor unter Bezugnahme auf die Fig. 10 30 beschrieben wurden.

Um das Ausführen einer Adressen-Operationsinstruktion zu erleichtern, gibt die IFU 60 AOU-Steuersignale, RF-Adressen und Konstante zu der AOU 64 während eines der beiden ISS-Zustände E oder M aus. Der dritte bzw. vierte RAM 220, 222 liefern ein erstes bzw. ein zweites Registerfile für die temporäre Adressenspeicherung. Individuelle Speicherstellen innerhalb des dritten und vierten RAM's 220, 222 werden gemäß den RF-Adressen 35 ausgewählt, die bei den jeweiligen Adressen-Auswähl-Eingang jedes RAM's empfangen werden. Das Laden des dritten und vierten RAM's 220, 222 wird durch die AOU-Steuersignale gesteuert, die das jeweilige RAM 220, 222 bei seinem Schreib-Freigabeeingang empfängt. Der Multiplexer 230 leitet selektiv Adressen, die durch den dritten und vierten RAM 220, 222 ausgegeben werden, zu der NIPAR 232, und zwar unter der Verwaltung der AOU-Steuersignale, die bei seinem Steuereingang empfangen werden. Der NIPAR 232 lädt eine Adresse, die 40 von dem Ausgang des Multiplexers 230 empfangen wird, und inkrementiert ihren Inhalt in Antwort auf das AOU-Steuersignal, das von seinem Steuereingang empfangen wird. Bei der bevorzugten Ausführungsform speichert der NIPAR 232 die Adresse der nächsten Programminstruktion, die auszuführen ist. Die arithmetische Einheit 234 führt arithmetische Operationen einschließlich Addition, Subtraktion, Inkrementation, Dekrementation bezüglich der Adressen durch, die von dem dritten und vierten RAM 220, 222 empfangen werden, und/oder 45 bezüglich des Inhalts des NIPAR 232. Der AOU-Crossbar-Schalter 200 leitet selektiv folgendes: 1) Adressen von dem Speicher 34 zu dem dritten und vierten RAM 220, 222; und 2) Ergebnisse der Adressenberechnungen, die durch die arithmetische Einheit 234 ausgegeben werden, zu dem Speicher 34 oder dem dritten und vierten RAM 220, 222. Der AOU-Crossbar-Schalter 200 führt eine bestimmte Leitwegoperation bzw. Wegermittlungsoperation 50 gemäß der AOU-Steuersignale durch, die bei seinem Steuereingang empfangen werden. Der Adressenmultiplexer 206 leitet selektiv die Adressen, die durch das dritte RAM 220 ausgegeben werden, Adressen, die durch das vierte RAM 222 ausgegeben werden, oder die Ergebnisse von Adressenberechnungen, die durch die arithmetische Einheit 234 ausgegeben werden, zu dem Adressenausgang des AOU, und zwar unter der Verwaltung des AOU-Steuersignals, das bei seinem Steuereingang empfangen wird.

Bei der bevorzugten Ausführungsform werden der dritte und vierte RAM 220, 222 jeweils realisiert, indem die 55 Datenspeicherschaltung, die innerhalb eines Satzes von CLBs vorliegt, verwendet wird. Der Multiplexer 230 und der Adressenmultiplexer 206 werden jeweils vorzugsweise realisiert, indem eine Datenauswahlschaltung verwendet wird, die innerhalb eines Satzes von CLBs vorhanden ist, und der NIPAR 232 wird vorzugsweise realisiert, indem die Datenspeicherschaltung, die innerhalb eines Satzes von CLBs vorliegt, verwendet wird. Die arithmetische Einheit 234 wird vorzugsweise realisiert, indem logische Funktionsgeneratoren und eine Schaltung, die auf mathematische Operationen innerhalb eines Satzes von CLBs ausgerichtet ist, verwendet wird. Schließlich wird der AOU-Crossbar-Schalter 200 vorzugsweise in einer zuvor beschriebenen Art und Weise 60 implementiert.

Es wird nun auf die Fig. 11B Bezug genommen. Dort ist ein Blockschaltbild einer zweiten exemplarischen Ausführungsform der AOU 66 gezeigt, die für die Implementation einer Innenschleifen-ISA konfiguriert ist. Vorzugsweise erfordert eine Innenschleifen-ISA eine Hardware zur Durchführung eines sehr beschränkten 65 Satzes von Adressenoperationen und eine Hardware zur Aufrechterhaltung wenigstens eines Quellenadressenzeichers und einer entsprechende Anzahl von Bestimmungsadressenzeichern. Arten von Innenschleifenverarbeitung, für die eine sehr beschränkte Anzahl von Adressenoperationen oder sogar eine einzige Adressenoperation

benötigt werden, beinhalten Block-, Raster- oder Serpentinenoperationen bezüglich der Bilddaten; Bit-Umkehroperationen; Operationen bezüglich zirkularer Pufferdaten; und Parsingoperationen bezüglich variabler Datenmengen. Hier wird eine einzige Adressenoperation, nämlich eine Inkrementoperation, betrachtet. Fachleute werden erkennen, daß Hardware, die Inkrementoperationen durchführt, ebenso inhärent in der Lage sein kann, Dekrementoperationen durchzuführen, wodurch eine zusätzliche Adressenoperationsfähigkeit bereitgestellt wird. Bei der zweiten beispielhaften Ausführungsform der AOU 66 umfaßt die Abspeicher-/Zähllogik 202 wenigstens ein Quellenadressenregister 202 mit einem Eingang, einem Ausgang und einem Steuereingang; wenigstens ein Bestimmungsadressenregister 254 mit einem Eingang, einem Ausgang und einem Steuereingang; und eine Datenauswähleinrichtung 250 mit einem Eingang, einem Steuereingang und einer Anzahl von Ausgängen, die gleich der gesamten Anzahl der vorliegenden Quellen- und Bestimmungsadressenregisters 252, 254 ist. Hier werden ein Einzelquellen-Adressenregister 252 und ein Einzelbestimmungs-Adressenregister 254 betrachtet, und somit hat die Datenauswähleinrichtung 250 einen ersten Ausgang und einen zweiten Ausgang. Die Adressenoperationslogik 204 umfaßt ein NIPAR 232 mit einem Eingang, einem Ausgang und einem Steuerausgang; einen Multiplexer 260 mit einer Anzahl von Eingängen, die gleich der Anzahl von Datenauswählausgängen ist, einen Steuereingang und einen Ausgang. Hier umfaßt der Multiplexer 260 einen ersten Eingang und einen zweiten Eingang. Der Adressenmultiplexer 206 umfaßt vorzugsweise einen Multiplexer mit einer Anzahl von Eingängen, die um 1 größer ist als die Anzahl von Datenauswählausgängen, einem Steuereingang und einem Ausgang. Somit umfaßt der Adressenmultiplexer 206 einen ersten Eingang, einen zweiten Eingang und einen dritten Eingang. Der AOU-Crossbar-Schalter 200 umfaßt vorzugsweise ein herkömmliches Crossbar-Schalt- 5
10
15
20
25
netzwerk mit bidirektionalen und unidirektionalen Crossbarverbindungen und mit den Eingängen und Ausgängen, die zuvor unter Bezugnahme auf die Fig. 10 beschrieben wurden. Eine wirksame Realisierung des AOU-Crossbar-Schalters 200 kann Multiplexer, Drei-Zustands-Puffer, eine CLB-basierte Logik, direkte Verdrahtung bzw. eine direkte Leitung oder sonstige Untersätze von derartigen Elementen enthalten, die durch rekonfigurierbare Verbindungen verbunden sind. Für eine Innenschleifen-ISA wird der AOU-Crossbar-Schalter 200 vorzugsweise realisiert, um eine parallele Adressenbewegung in einer minimal möglichen Zeit zu maximieren, während ebenso eine minimale Anzahl von einzigen bzw. eindeutigen Adressen-Bewegungs-Crossbar-Verbindungen bereitgestellt werden, um Innenschleifen-Adressenoperationen zu unterstützen.

Der Eingang der Datenauswähleinrichtung 250 ist mit dem Ausgang des AOU-Crossbar-Schalters 200 verbunden. Der erste bzw. zweite Ausgang der Datenauswähleinrichtung 250 sind mit dem Eingang des Quellen- 30
adressenregisters 252 bzw. mit dem Eingang des Bestimmungsadressenregisters 254 verbunden. Die Steuerein- 35
gänge des Quellenadressenregisters 252 und des Bestimmungsadressenregisters 254 sind verbunden, um AOU- 40
Steuersignale über die zweite Steuerleitung 72 zu empfangen. Der Ausgang des Quellenadressenregisters 252 ist mit dem ersten Eingang des Multiplexers 260 und dem ersten Eingang des Adressenmultiplexers 206 verbunden. In ähnlicher Weise ist der Ausgang des Bestimmungsregisters 254 mit dem zweiten Eingang des Multiplexers 254 und dem zweiten Eingang des Adressenmultiplexers 206 verbunden. Der Eingang des NIPAR 232 ist mit dem Ausgang des Multiplexers 260 verbunden, der Steuereingang des NIPAR 232 ist verbunden, um AOU-Steuersignale über die zweite Steuerleitung zu empfangen, und der Ausgang des NIPAR 232 ist sowohl mit dem Adressen-Rückföhreingang bzw. Adressen-Feedback-Eingang des AOU-Crossbar-Schalters 200 als auch mit dem dritten Eingang des Adressenmultiplexers 206 verbunden. Die Verbindungen zu den übrigen Eingängen und Ausgängen des AOU-Crossbar-Schalters 200 sind mit den zuvor unter Bezugnahme auf die Fig. 10 beschriebenen identisch.

Im Betrieb leitet die Datenauswähleinrichtung 250 Adressen, die von dem AOU-Crossbar-Schalter empfan- 45
gen werden, zu dem Quellenadressenregister 252 oder dem Bestimmungsadressenregister 254 gemäß den RF-Adressen, die bei ihrem Steuereingang empfangen werden. Das Quellenadressenregister 252 lädt eine 50
Adresse, die an ihrem Eingang vorliegt, als Antwort auf das AOU-Steuersignal, das bei seinem Steuereingang vorliegt. Das Bestimmungsadressenregister 254 lädt eine Adresse, die bei seinem Eingang vorliegt, in einer 55
analogen Art und Weise. Der Multiplexer 260 leitet eine Adresse, die von dem Quellenadressenregister 252 oder dem Bestimmungsadressenregister 254 empfangen wird, zu dem Eingang des NIPAR 232 gemäß den AOU-Steuersignalen, die bei seinem Steuereingang empfangen werden. Das NIPAR 232 lädt eine Adresse, die bei seinem 60
Eingang vorliegt, inkrementiert ihren Inhalt oder dekrementiert ihren Inhalt in Antwort auf die AOU-Steuersignale, die bei seinem Steuereingang empfangen werden. Der AOU-Crossbar-Schalter 200 leitet selektiv folgen- 65
des: 1) Adressen von dem Speicher 34 zu der Datenauswähleinrichtung 250; und 2) die Inhalte des NIPAR 232 zu dem Speicher 34 oder der Datenauswähleinrichtung 250. Der AOU-Crossbar-Schalter 200 führt eine bestimmte Leitwegoperation bzw. Wegeermittlungsoperation gemäß den AOU-Steuersignalen durch, die bei seinem Eingang empfangen werden. Der Adressenmultiplexer 206 leitet selektiv die Inhalte des Quellenadressenregisters 252, des Bestimmungsadressenregisters 254 oder des NIPAR 232 zu dem Adressenausgang des AOU's, und zwar unter der Verwaltung der AOU-Steuersignale, die aus einem Steuereingang empfangen werden.

Bei der bevorzugten Ausführungsform sind sowohl das Quellenadressenregister 252 als auch das Bestim- 70
mungsadressenregister 254 verwirklicht, indem die Datenspeicherschaltung verwendet wird, die innerhalb eines Satzes von CLBs vorliegt. Das NIPAR 232 wird vorzugsweise verwirklicht, indem eine Inkrement-/Dekrement- 75
logik und Flipflops innerhalb eines Satzes von CLBs verwendet werden. Die Datenauswähleinrichtung 250, der Multiplexer 230 und der Adressenmultiplexer 206 werden jeweils vorzugsweise verwirklicht, indem eine Datenauswahlschaltung verwendet wird, die innerhalb eines Satzes von CLBs vorliegt. Schließlich wird der AOU-Crossbar-Schalter 200 vorzugsweise in der zuvor für einen Innenschleifen-ISA beschriebenen Art und Weise 80
verwirklicht. Fachleute werden erkennen, daß es bei bestimmten Anwendungen vorteilhaft sein kann, eine ISA zu verwenden, die sich auf eine Innenschleifen-AOU-Konfiguration mit einer Außenschleifen-DOU-Konfiguration oder umgekehrt verläßt. Zum Beispiel würde eine assoziative String-Such-ISA vorteilhaft eine Innenschleifen-DOU-Konfiguration mit einer Außenschleifen-AOU-Konfiguration nutzen. Gemäß einem anderen Beispiel 85

würde eine ISA zur Durchführung von Histogrammoperationen vorteilhaft eine Außenschleifen-DOU-Konfiguration mit einer Innenschleifen-AOU-Konfiguration nutzen.

Finite bzw. endliche rekonfigurierbare Hardware-Systemelemente müssen zwischen jedem Element der DRPU 32 zugewiesen werden. Weil die rekonfigurierbaren Hardware-Systemelemente in ihrer Anzahl beschränkt sind, beeinflußt bzw. beeinträchtigt die Art und Weise, in der sie der IFU 60 zugeordnet sind, z. B. den maximalen Rechenleistungsfähigkeitspegel, der durch die DOU 62 und die AOU 64 erreichbar ist. Die Art und Weise, in der die rekonfigurierbaren Hardware-Systemelemente zwischen der IFU 60, der DOU 62 und der AOU 64 zugewiesen sind, variiert gemäß der Art der ISA, die zu jedem gegebenen Augenblick zu implementieren ist. Wenn die ISA-Komplexität zunimmt, müssen mehr rekonfigurierbare Hardware-Systemelemente der IFU 60 zugewiesen werden, um die zunehmend komplexen Decodier- und Steueroperationen zu erleichtern, was dazu führt, daß weniger rekonfigurierbare Hardware-Systemelemente zwischen der DOU 62 und der AOU 64 verfügbar bleiben. Somit nimmt die maximale Rechenleistung, die von der DOU 62 und der AOU 64 erreicht werden kann, mit der ISA-Komplexität ab. Im allgemeinen wird eine Außenschleifen-ISA viel mehr Instruktionen aufweisen als eine Innenschleifen-ISA, und deshalb wird ihre Realisierung in Termen der Decodier- und Steuerschaltung beträchtlich komplexer sein. Zum Beispiel würde eine Außenschleifen-ISA, die einen Allzweck-64-Bit-Prozessor festlegt, viel mehr Instruktionen aufweisen als eine Innenschleifen-ISA, die nur auf Datenkomprimierung ausgerichtet ist.

Nimmt man nun Bezug auf die Fig. 12A, so ist ein Diagramm gezeigt, das eine exemplarische Zuweisung von rekonfigurierbaren Hardware-Systemelementen zwischen der IFU 60, der DOU 62 und der AOU 64 für eine Außenschleifen-ISA zeigt. Bei der beispielhaften Zuweisung von rekonfigurierbaren Hardware-Systemelementen für die Außenschleifen-ISA werden der IFU 60, der DOU 62 und der AOU 64 jeweils ein Drittel der verfügbaren rekonfigurierbaren Hardware-Systemelemente bzw. Hardware-Ressourcen zugewiesen. Für den Fall, daß die DRPU 32 rekonfiguriert werden soll, um eine Innenschleifen-ISA zu verwirklichen, werden weniger rekonfigurierbare Hardware-Systemelemente benötigt, um die IFU 60 und die AOU 64 zu verwirklichen, und zwar aufgrund der beschränkten Anzahl von Instruktionen und Typen von Adressenoperationen, die durch eine Innenschleifen-ISA unterstützt werden. Nimmt man nun Bezug auf die Fig. 12B, so ist ein Diagramm gezeigt, daß eine beispielhafte Zuweisung von rekonfigurierbaren Hardware-Systemelementen bzw. Hardware-Ressourcen zwischen der IFU 60, der DOU 62 und der AOU 64 für eine Innenschleifen-ISA zeigt. Bei der beispielhaften Zuweisung von rekonfigurierbaren Hardware-Systemelementen für die Innenschleifen-ISA wird die IFU 60 verwirklicht, indem ungefähr 5—10% der rekonfigurierbaren Hardware-Systemelemente verwendet werden, und die AOU 64 wird verwirklicht, indem ungefähr 10—25% der rekonfigurierbaren Hardware-Systemelemente verwendet werden. Somit verbleiben ungefähr 70—80% der rekonfigurierbaren Hardware-Systemelemente für die Realisierung der DOU 62 verfügbar. Dies wiederum bedeutet, daß die interne Struktur der DOU 62, die mit der Innenschleifen-ISA in Zusammenhang steht bzw. dieser zugeordnet ist, komplexer sein kann und deshalb eine beträchtlich höhere Leistungsfähigkeit anbietet als die interne Struktur der DOU 62, die mit der Außenschleifen-ISA im Zusammenhang steht bzw. dieser zugeordnet ist.

Fachleute werden erkennen, daß die DRPU 32 in einer alternativen Ausführungsform entweder die DOU 62 oder die AOU 64 ausschließen kann. Zum Beispiel kann in einer alternativen Ausführungsform die DRPU 32 nicht eine AOU 64 beinhalten. Die DOU 62 würde dann für die Durchführungsoperationen bezüglich sowohl der Daten als auch der Adressen verantwortlich sein. Unabhängig von der bestimmten DRPU-Ausführung, die betrachtet wird, muß eine finite bzw. endliche Zahl von rekonfigurierbaren Hardwarequellen zugewiesen werden, um die Elemente der DRPU 32 zu realisieren. Die rekonfigurierbaren Hardware-Systemelemente werden vorzugsweise derart zugewiesen, daß eine optimale Leistungsfähigkeit oder eine beinahe optimale Leistungsfähigkeit bezüglich der gegenwärtig betrachteten ISA relativ zu dem gesamten Raum von verfügbaren rekonfigurierbaren Hardware-Systemelementen erreicht wird. Fachleute werden erkennen, daß die detaillierte Struktur eines jeden Elements der IFU 60, der DOU 62 und der AOU 64 nicht auf die Ausführungsformen beschränkt ist, die oben beschrieben wurden. Für eine gegebene ISA ist der entsprechende Konfigurationsdatensatz vorzugsweise derart festgelegt, daß die interne Struktur eines jeden Elements innerhalb der IFU 60, der DOU 62 und der AOU 64 die Rechenleistungsfähigkeit relativ zu den verfügbaren rekonfigurierbaren Hardware-Systemelementen maximiert.

Nimmt man nun Bezug auf die Fig. 13, so ist ein Blockdiagramm einer bevorzugten Ausführungsform einer T-Maschine 14 gezeigt. Die T-Maschine 14 umfaßt eine zweite lokale Zeitbasisseinheit 300, eine gemeinsame Schnittstellen- und Steuereinheit 302 und einen Satz von Verbindungs-I/O-Einheiten 304. Die zweite lokale Zeitbasisseinheit 300 weist einen Zeitsteuereingang auf, der den Master-Zeitsteuereingang der T-Maschine ausbildet. Die gemeinsame Schnittstellen- und Steuereinheit 302 umfaßt einen Zeitsteuereingang, der mit einem Zeitsteuerausgang der zweiten lokalen Zeitbasisseinheit 300 über eine zweite Zeitsteuersignalleitung 310 verbunden ist, einen Adressenausgang, der mit der Adressleitung 44 verbunden ist, einen ersten bidirektionalen Datenport, der mit der Speicher-I/O-Leitung 46 verbunden ist, einen bidirektionalen Steueroport, der mit der externen Steuerleitung 48 verbunden ist, und einen zweiten bidirektionalen Datenport, der mit einem bidirektionalen Datenport einer jeden Verbindungs-I/O-Einheit 304, die vorliegt, über eine Nachrichtenübertragungsleitung 312 verbunden ist. Jede Verbindungs-I/O-Einheit 304 umfaßt einen Eingang, der mit der GPIM 16 über eine Nachrichteneingangsleitung 314 verbunden ist, und einen Ausgang, der mit der GPIM 16 über eine Nachrichtenausgangsleitung 316 verbunden ist.

Die zweite lokale Zeitbasisseinheit 300 innerhalb der T-Maschine 14 empfängt das Master-Zeitsteuersignal von der Master-Zeitbasisseinheit 22 und erzeugt ein zweites lokals Zeitsteuersignal. Die zweite lokale Zeitsteuereinheit 300 liefert das zweite lokale Zeitsteuersignal zu der gemeinsamen Schnittstellen- und Steuereinheit 302, dadurch wird eine Zeitsteuerreferenz für die T-Maschine 14 geliefert, auf der sie beruht. Vorzugsweise ist das zweite lokale Zeitsteuersignal phasensynchronisiert mit dem Master-Zeitsteuersignal. Innerhalb des Systems 10

operiert bzw. arbeitet jede zweite lokale Zeitbasiseinheit 300 der T-Maschine vorzugsweise mit einer identischen Frequenz. Fachleute werden erkennen, daß bei einer alternativen Ausführungsform eine oder mehrere zweite lokale Zeitbasiseinheiten 300 bei verschiedenen Frequenzen arbeiten können. Die zweite lokale Zeitbasiseinheit 300 wird vorzugsweise realisiert, indem eine herkömmliche phasensynchronisierte Frequenzumwandlungsschaltung verwendet wird, einschließlich einer CLB-basierten phasensynchronisierten Detektionsschaltung. Fachleute werden erkennen, daß bei einer alternativen Ausführungsform die zweite lokale Zeitbasiseinheit 300 als ein Abschnitt eines Taktverteilungsbaumes realisiert werden könnte.

Die gemeinsame Schnittstellen- und Steuereinheit 302 verwaltet den Transfer von Nachrichten zwischen ihrer entsprechenden S-Maschine 12 und einer spezifizierten Verbindungs-I/O-Einheit 304, wo eine Nachricht ein Kommando bzw. einen Befehl und mögliche Daten enthält. Bei der bevorzugten Ausführungsform kann die spezifizierte Verbindungs-I/O-Einheit 304 innerhalb jeglicher T-Maschine 14 oder I/O-T-Maschine 18 sich innerhalb oder außerhalb bezüglich des Systems 10 befinden. Bei der vorliegenden Erfindung ist jeder Verbindungs-I/O-Einheit 304 vorzugsweise eine Verbindungsadresse zugeordnet, die einzig die Verbindungs-I/O-Einheit 304 identifiziert. Die Verbindungsadressen zur Verbindung der I/O-Einheiten 304 innerhalb einer gegebenen T-Maschine werden in dem entsprechenden Architekturbeschreibungsspeicher 101 der S-Maschine gespeichert.

Die gemeinsame Schnittstellen- und Steuereinheit 302 empfängt Daten bzw. Befehle von ihrer entsprechenden S-Maschine 12 über die Speicher-I/O-Leitung 46 bzw. die externe Steuersignalleitung 48. Vorzugsweise beinhaltet jeder empfangene Befehl eine Zielverbindungsadresse und einen Befehlscode bzw. Kommandocode, der einen bestimmten Operationstyp spezifiziert, der durchgeführt werden soll. In der bevorzugten Ausführungsform beinhalten die Operationstypen, die einzig bzw. eindeutig durch die Kommandocodes bzw. Befehlscodes identifiziert werden, folgendes: 1) Datenleseoperationen; 2) Datenschreiboperationen; und 3) Interruptsignaltransfer einschließlich Rekonfigurations-Interruptsignaltransfer. Die Zielverbindungsadressen identifizieren eine Zielverbindungs-I/O-Einheit 304, zu der Daten und Befehle übertragen werden soll. Vorzugsweise überträgt die gemeinsame Schnittstellen- und Steuereinheit 302 jeden Befehl und jegliche in Beziehung stehende Daten als einen Satz von Nachrichten auf Paketbasis in einer herkömmlichen Art und Weise, und zwar wo jede Nachricht die Zielverbindungsadresse und den Befehlscode enthält.

Zusätzlich zum Empfang der Daten und Befehle von ihrer entsprechenden S-Maschine 12 empfängt die gemeinsame Schnittstellen- und Steuereinheit 302 Nachrichten bzw. Meldungen von jeder der Verbindungs-I/O-Einheiten 304, die mit der Nachrichtenübertragungsleitung bzw. Nachrichtentransferleitung 312 verbunden sind. Bei der bevorzugten Ausführungsform wandelt die gemeinsame Schnittstellen- und Steuereinheit 302 eine Gruppe von in Beziehung stehenden Nachrichten in eine einzige Befehls- und Datensequenz um. Falls der Befehl auf die DRPU 32 innerhalb ihrer entsprechenden S-Maschine 12 gerichtet ist, gibt die gemeinsame Schnittstellen- und Steuereinheit 302 den Befehl über die externe Steuersignalleitung 48 aus. Falls der Befehl auf den Speicher 34 innerhalb ihrer entsprechenden S-Maschine 12 gerichtet ist, gibt die gemeinsame Schnittstellen- und Steuereinheit 302 ein entsprechendes Speichersteuersignal über die externe Steuersignalleitung 48 und ein Speicheradressensignal über die Speicheradressleitung 44 aus. Daten werden über die Speicher-I/O-Leitung 46 übertragen. Bei der bevorzugten Ausführungsform umfaßt die gemeinsame Schnittstellen- und Steuereinheit 302 eine Schaltung auf CLB-Basis, um Operationen analog zu jenen, die durch eine konventionelle SCI-Schaltungseinheit durchgeführt werden, wie sie durch den ANSI/IEEE-Standard 1596—1992 definiert ist, zu realisieren.

Jede Verbindungs-I/O-Einheit 304 empfängt Nachrichten von der gemeinsamen Schnittstellen- und Steuereinheit 302 und überträgt Nachrichten zu anderen Verbindungs-I/O-Einheiten 304 über die GPIM 16, und zwar unter Verwaltung der Steuersignale, die von der gemeinsamen Schnittstellen- und Steuereinheit 302 empfangen werden. Bei der bevorzugten Ausführungsform basiert die Verbindungs-I/O-Einheit 304 auf einem SCI-Knoten, wie durch den ANSI/IEEE-Standard 1596—1992 definiert ist. Nimmt man nun Bezug auf die Fig. 14, so ist ein Blockdiagramm einer bevorzugten Ausführungsform einer Verbindungs-I/O-Einheit 304 gezeigt. Die Verbindungs-I/O-Einheit 304 umfaßt einen Adressendecoder 302, einen Eingangs-FIFO-Puffer 322, einen Bypass-FIFO-Puffer 324 und einen Ausgangs-FIFO-Puffer 326 und einen Multiplexer 328. Der Adressendecoder 328 umfaßt einen Eingang, der den Eingang der Verbindungs-I/O-Einheit ausbildet, einen ersten Ausgang, der mit dem Eingangs-FIFO 322 verbunden ist, und einen zweiten Ausgang, der mit dem Bypass-FIFO 324 verbunden ist. Der Eingangs-FIFO 322 umfaßt einen Ausgang, der mit der Nachrichtentransferleitung 312 zur Übertragung von Nachrichten zu der gemeinsamen Schnittstellen- und Steuereinheit 302 verbunden ist. Der Ausgangs-FIFO 326 umfaßt einen Eingang, der mit der Nachrichtentransferleitung 312 zum Empfang von Nachrichten von der gemeinsamen Schnittstellen- und Steuereinheit 302 verbunden ist, und einen Ausgang, der mit dem ersten Eingang des Multiplexers 328 verbunden ist. Der Bypass-FIFO 324 umfaßt einen Ausgang, der mit einem zweiten Eingang des Multiplexers 328 verbunden ist. Schließlich umfaßt der Multiplexer 328 einen Steuereingang, der mit der Nachrichtentransferleitung 312 verbunden ist, und einen Ausgang, der den Ausgang der Verbindungs-I/O-Einheit ausbildet.

Die Verbindungs-I/O-Einheit 304 empfängt Nachrichten bei dem Eingang des Adressendecoders 320. Der Adressendecoder 320 bestimmt, ob die Zielverbindungsadresse, die in einer empfangenen Nachricht spezifiziert ist, identisch mit der Verbindungsadresse der Verbindungs-I/O-Einheit 304 ist, in der sie sich befindet. Falls dem so ist, leitet der Adressendecoder 320 die Nachricht zu dem Eingangs-FIFO 322. Ansonsten leitet der Adressendecoder 320 die Nachricht zu dem Bypass-FIFO 324. Bei der bevorzugten Ausführungsform umfaßt der Adressendecoder 320 einen Decoder und eine Datenauswähleinrichtung, die verwirklicht ist, indem IOBs und CLBs verwendet werden.

Bei dem Eingangs-FIFO 322 handelt es sich um einen konventionellen FIFO-Puffer, der Nachrichten, die bei seinem Eingang empfangen werden, zu der Nachrichtenübertragungsleitung 312 überträgt. Sowohl der Bypass-

5 FIFO 324 als auch der Ausgangs-FIFO 326 sind herkömmliche FIFO-Puffer, die Nachrichten, die bei deren Eingängen empfangen werden, zu dem Multiplexer 328 übertragen. Der Multiplexer 328 ist ein herkömmlicher Multiplexer, der entweder eine Nachricht, die von dem Bypaß-FIFO 324 erhalten wird, oder eine Nachricht, die von dem Ausgangs-FIFO 326 erhalten wird, zu der GPIM 16 in Übereinstimmung mit einem Steuersignal, das bei seinem Steuereingang empfangen wird, leitet. Bei der bevorzugten Ausführungsform werden sowohl der Eingangs-FIFO 322 als auch der Bypaß-FIFO 324 und der Ausgangs-FIFO 326 realisiert, indem ein Satz von CLBs verwendet wird. Der Multiplexer 328 wird vorzugsweise realisiert, indem ein Satz von CLBs und IOBs verwendet wird.

10 Es wird nun auf die Fig. 15 Bezug genommen. Dort ist ein Blockdiagramm einer bevorzugten Ausführungsform einer I/O-T-Maschine 18 gezeigt. Die I/O-T-Maschine 18 umfaßt eine dritte lokale Zeitbasiseinheit 360, eine allgemeine Kunden-Schnittstellen- und Steuereinheit 362 und eine Verbindungs-I/O-Einheit 304. Die dritte lokale Zeitbasiseinheit 360 umfaßt eine Zeitbasiseinheit, die den Master-Zeitsteuereingang der I/O-T-Maschine ausbildet. Die Verbindungs-I/O-Einheit 304 umfaßt einen Eingang, der mit der GPIM 16 über eine Nachrichteneingangsleitung 314 verbunden ist, und einen Ausgang, der mit der GPIM 16 über eine Nachrichtenausgangsleitung 316 verbunden ist. Die gemeinsame Kunden-Schnittstellen- und Steuereinheit 362 umfaßt vorzugsweise einen Zeitsteuereingang, der mit einem Zeitsteuerausgang der dritten lokalen Zeitbasiseinheit 360 über eine dritte Zeitsteuersignalleitung 370 verbunden ist, einen ersten bidirektionalen Datenport, der mit einem bidirektionalen Datenport der Verbindungs-I/O-Einheit 304 verbunden ist, und einen Satz von Verbindungen zu einer I/O-Vorrichtung 20. Bei der bevorzugten Ausführungsform beinhaltet der Satz von Verbindungen zu der I/O-Vorrichtung 20 einen zweiten bidirektionalen Datenport, der mit einem bidirektionalen Datenport der I/O-Vorrichtung 20 verbunden ist, und einen Adressenausgang, der mit einem Adresseneingang von der I/O-Vorrichtung 20 verbunden ist, und einen bidirektionalen Steuerport, der mit einem bidirektionalen Steuerport der I/O-Vorrichtung 20 verbunden ist. Fachleute werden leicht erkennen, daß die Verbindungen zu der I/O-Vorrichtung 20 von dem Typ der I/O-Vorrichtung 20 abhängen, mit der die gemeinsam Kunden-Schnittstellen- und Steuereinheit 362 verbunden ist.

15 20 25 30 35 40 45 50 55 60 65 70 75 80 85 90 95 100 105 110 115 120 125 130 135 140 145 150 155 160 165 170 175 180 185 190 195 200 205 210 215 220 225 230 235 240 245 250 255 260 265 270 275 280 285 290 295 300 305 310 315 320 325 330 335 340 345 350 355 360 365 370 375 380 385 390 395 400 405 410 415 420 425 430 435 440 445 450 455 460 465 470 475 480 485 490 495 500 505 510 515 520 525 530 535 540 545 550 555 560 565 570 575 580 585 590 595 600 605 610 615 620 625 630 635 640 645 650 655 660 665 670 675 680 685 690 695 700 705 710 715 720 725 730 735 740 745 750 755 760 765 770 775 780 785 790 795 800 805 810 815 820 825 830 835 840 845 850 855 860 865 870 875 880 885 890 895 900 905 910 915 920 925 930 935 940 945 950 955 960 965 970 975 980 985 990 995 1000 1005 1010 1015 1020 1025 1030 1035 1040 1045 1050 1055 1060 1065 1070 1075 1080 1085 1090 1095 1100 1105 1110 1115 1120 1125 1130 1135 1140 1145 1150 1155 1160 1165 1170 1175 1180 1185 1190 1195 1200 1205 1210 1215 1220 1225 1230 1235 1240 1245 1250 1255 1260 1265 1270 1275 1280 1285 1290 1295 1300 1305 1310 1315 1320 1325 1330 1335 1340 1345 1350 1355 1360 1365 1370 1375 1380 1385 1390 1395 1400 1405 1410 1415 1420 1425 1430 1435 1440 1445 1450 1455 1460 1465 1470 1475 1480 1485 1490 1495 1500 1505 1510 1515 1520 1525 1530 1535 1540 1545 1550 1555 1560 1565 1570 1575 1580 1585 1590 1595 1600 1605 1610 1615 1620 1625 1630 1635 1640 1645 1650 1655 1660 1665 1670 1675 1680 1685 1690 1695 1700 1705 1710 1715 1720 1725 1730 1735 1740 1745 1750 1755 1760 1765 1770 1775 1780 1785 1790 1795 1800 1805 1810 1815 1820 1825 1830 1835 1840 1845 1850 1855 1860 1865 1870 1875 1880 1885 1890 1895 1900 1905 1910 1915 1920 1925 1930 1935 1940 1945 1950 1955 1960 1965 1970 1975 1980 1985 1990 1995 2000 2005 2010 2015 2020 2025 2030 2035 2040 2045 2050 2055 2060 2065 2070 2075 2080 2085 2090 2095 2100 2105 2110 2115 2120 2125 2130 2135 2140 2145 2150 2155 2160 2165 2170 2175 2180 2185 2190 2195 2200 2205 2210 2215 2220 2225 2230 2235 2240 2245 2250 2255 2260 2265 2270 2275 2280 2285 2290 2295 2300 2305 2310 2315 2320 2325 2330 2335 2340 2345 2350 2355 2360 2365 2370 2375 2380 2385 2390 2395 2400 2405 2410 2415 2420 2425 2430 2435 2440 2445 2450 2455 2460 2465 2470 2475 2480 2485 2490 2495 2500 2505 2510 2515 2520 2525 2530 2535 2540 2545 2550 2555 2560 2565 2570 2575 2580 2585 2590 2595 2600 2605 2610 2615 2620 2625 2630 2635 2640 2645 2650 2655 2660 2665 2670 2675 2680 2685 2690 2695 2700 2705 2710 2715 2720 2725 2730 2735 2740 2745 2750 2755 2760 2765 2770 2775 2780 2785 2790 2795 2800 2805 2810 2815 2820 2825 2830 2835 2840 2845 2850 2855 2860 2865 2870 2875 2880 2885 2890 2895 2900 2905 2910 2915 2920 2925 2930 2935 2940 2945 2950 2955 2960 2965 2970 2975 2980 2985 2990 2995 3000 3005 3010 3015 3020 3025 3030 3035 3040 3045 3050 3055 3060 3065 3070 3075 3080 3085 3090 3095 3100 3105 3110 3115 3120 3125 3130 3135 3140 3145 3150 3155 3160 3165 3170 3175 3180 3185 3190 3195 3200 3205 3210 3215 3220 3225 3230 3235 3240 3245 3250 3255 3260 3265 3270 3275 3280 3285 3290 3295 3300 3305 3310 3315 3320 3325 3330 3335 3340 3345 3350 3355 3360 3365 3370 3375 3380 3385 3390 3395 3400 3405 3410 3415 3420 3425 3430 3435 3440 3445 3450 3455 3460 3465 3470 3475 3480 3485 3490 3495 3500 3505 3510 3515 3520 3525 3530 3535 3540 3545 3550 3555 3560 3565 3570 3575 3580 3585 3590 3595 3600 3605 3610 3615 3620 3625 3630 3635 3640 3645 3650 3655 3660 3665 3670 3675 3680 3685 3690 3695 3700 3705 3710 3715 3720 3725 3730 3735 3740 3745 3750 3755 3760 3765 3770 3775 3780 3785 3790 3795 3800 3805 3810 3815 3820 3825 3830 3835 3840 3845 3850 3855 3860 3865 3870 3875 3880 3885 3890 3895 3900 3905 3910 3915 3920 3925 3930 3935 3940 3945 3950 3955 3960 3965 3970 3975 3980 3985 3990 3995 4000 4005 4010 4015 4020 4025 4030 4035 4040 4045 4050 4055 4060 4065 4070 4075 4080 4085 4090 4095 4100 4105 4110 4115 4120 4125 4130 4135 4140 4145 4150 4155 4160 4165 4170 4175 4180 4185 4190 4195 4200 4205 4210 4215 4220 4225 4230 4235 4240 4245 4250 4255 4260 4265 4270 4275 4280 4285 4290 4295 4300 4305 4310 4315 4320 4325 4330 4335 4340 4345 4350 4355 4360 4365 4370 4375 4380 4385 4390 4395 4400 4405 4410 4415 4420 4425 4430 4435 4440 4445 4450 4455 4460 4465 4470 4475 4480 4485 4490 4495 4500 4505 4510 4515 4520 4525 4530 4535 4540 4545 4550 4555 4560 4565 4570 4575 4580 4585 4590 4595 4600 4605 4610 4615 4620 4625 4630 4635 4640 4645 4650 4655 4660 4665 4670 4675 4680 4685 4690 4695 4700 4705 4710 4715 4720 4725 4730 4735 4740 4745 4750 4755 4760 4765 4770 4775 4780 4785 4790 4795 4800 4805 4810 4815 4820 4825 4830 4835 4840 4845 4850 4855 4860 4865 4870 4875 4880 4885 4890 4895 4900 4905 4910 4915 4920 4925 4930 4935 4940 4945 4950 4955 4960 4965 4970 4975 4980 4985 4990 4995 5000 5005 5010 5015 5020 5025 5030 5035 5040 5045 5050 5055 5060 5065 5070 5075 5080 5085 5090 5095 5100 5105 5110 5115 5120 5125 5130 5135 5140 5145 5150 5155 5160 5165 5170 5175 5180 5185 5190 5195 5200 5205 5210 5215 5220 5225 5230 5235 5240 5245 5250 5255 5260 5265 5270 5275 5280 5285 5290 5295 5300 5305 5310 5315 5320 5325 5330 5335 5340 5345 5350 5355 5360 5365 5370 5375 5380 5385 5390 5395 5400 5405 5410 5415 5420 5425 5430 5435 5440 5445 5450 5455 5460 5465 5470 5475 5480 5485 5490 5495 5500 5505 5510 5515 5520 5525 5530 5535 5540 5545 5550 5555 5560 5565 5570 5575 5580 5585 5590 5595 5600 5605 5610 5615 5620 5625 5630 5635 5640 5645 5650 5655 5660 5665 5670 5675 5680 5685 5690 5695 5700 5705 5710 5715 5720 5725 5730 5735 5740 5745 5750 5755 5760 5765 5770 5775 5780 5785 5790 5795 5800 5805 5810 5815 5820 5825 5830 5835 5840 5845 5850 5855 5860 5865 5870 5875 5880 5885 5890 5895 5900 5905 5910 5915 5920 5925 5930 5935 5940 5945 5950 5955 5960 5965 5970 5975 5980 5985 5990 5995 6000 6005 6010 6015 6020 6025 6030 6035 6040 6045 6050 6055 6060 6065 6070 6075 6080 6085 6090 6095 6100 6105 6110 6115 6120 6125 6130 6135 6140 6145 6150 6155 6160 6165 6170 6175 6180 6185 6190 6195 6200 6205 6210 6215 6220 6225 6230 6235 6240 6245 6250 6255 6260 6265 6270 6275 6280 6285 6290 6295 6300 6305 6310 6315 6320 6325 6330 6335 6340 6345 6350 6355 6360 6365 6370 6375 6380 6385 6390 6395 6400 6405 6410 6415 6420 6425 6430 6435 6440 6445 6450 6455 6460 6465 6470 6475 6480 6485 6490 6495 6500 6505 6510 6515 6520 6525 6530 6535 6540 6545 6550 6555 6560 6565 6570 6575 6580 6585 6590 6595 6600 6605 6610 6615 6620 6625 6630 6635 6640 6645 6650 6655 6660 6665 6670 6675 6680 6685 6690 6695 6700 6705 6710 6715 6720 6725 6730 6735 6740 6745 6750 6755 6760 6765 6770 6775 6780 6785 6790 6795 6800 6805 6810 6815 6820 6825 6830 6835 6840 6845 6850 6855 6860 6865 6870 6875 6880 6885 6890 6895 6900 6905 6910 6915 6920 6925 6930 6935 6940 6945 6950 6955 6960 6965 6970 6975 6980 6985 6990 6995 7000 7005 7010 7015 7020 7025 7030 7035 7040 7045 7050 7055 7060 7065 7070 7075 7080 7085 7090 7095 7100 7105 7110 7115 7120 7125 7130 7135 7140 7145 7150 7155 7160 7165 7170 7175 7180 7185 7190 7195 7200 7205 7210 7215 7220 7225 7230 7235 7240 7245 7250 7255 7260 7265 7270 7275 7280 7285 7290 7295 7300 7305 7310 7315 7320 7325 7330 7335 7340 7345 7350 7355 7360 7365 7370 7375 7380 7385 7390 7395 7400 7405 7410 7415 7420 7425 7430 7435 7440 7445 7450 7455 7460 7465 7470 7475 7480 7485 7490 7495 7500 7505 7510 7515 7520 7525 7530 7535 7540 7545 7550 7555 7560 7565 7570 7575 7580 7585 7590 7595 7600 7605 7610 7615 7620 7625 7630 7635 7640 7645 7650 7655 7660 7665 7670 7675 7680 7685 7690 7695 7700 7705 7710 7715 7720 7725 7730 7735 7740 7745 7750 7755 7760 7765 7770 7775 7780 7785 7790 7795 7800 7805 7810 7815 7820 7825 7830 7835 7840 7845 7850 7855 7860 7865 7870 7875 7880 7885 7890 7895 7900 7905 7910 7915 7920 7925 7930 7935 7940 7945 7950 7955 7960 7965 7970 7975 7980 7985 7990 7995 8000 8005 8010 8015 8020 8025 8030 8035 8040 8045 8050 8055 8060 8065 8070 8075 8080 8085 8090 8095 8100 8105 8110 8115 8120 8125 8130 8135 8140 8145 8150 8155 8160 8165 8170 8175 8180 8185 8190 8195 8200 8205 8210 8215 8220 8225 8230 8235 8240 8245 8250 8255 8260 8265 8270 8275 8280 8285 8290 8295 8300 8305 8310 8315 8320 8325 8330 8335 8340 8345 8350 8355 8360 8365 8370 8375 8380 8385 8390 8395 8400 8405 8410 8415 8420 8425 8430 8435 8440 8445 8450 8455 8460 8465 8470 8475 8480 8485 8490 8495 8500 8505 8510 8515 8520 8525 8530 8535 8540 8545 8550 8555 8560 8565 8570 8575 8580 8585 8590 8595 8600 8605 8610 8615 8620 8625 8630 8635 8640 8645 8650 8655 8660 8665 8670 8675 8680 8685 8690 8695 8700 8705 8710 8715 8720 8725 8730 8735 8740 8745 8750 8755 8760 8765 8770 8775 8780 8785 8790 8795 8800 8805 8810 8815 8820 8825 8830 8835 8840 8845 8850 8855 8860 8865 8870 8875 8880 8885 8890 8895 8900 8905 8910 8915 8920 8925 8930 8935 8940 8945 8950 8955 8960 8965 8970 8975 8980 8985 8990 8995 9000 9005 9010 9015 9020 9025 9030 9035 9040 9045 9050 9055 9060 9065 9070 9075 9080 9085 9090 9095 9100 9105 9110 9115 9120 9125 9130 9135 9140 9145 9150 9155 9160 9165 9170 9175 9180 9185 9190 9195 9200 9205 9210 9215 9220 9225 9230 9235 9240 9245 9250 9255 9260 9265 9270 9275 9280 9285 9290 9295 9300 9305 9310 9315 9320 9325 9330 9335 9340 9345 9350 9355 9360 9365 9370 9375 9380 9385 9390 9395 9400 9405 9410 9415 9420 9425 9430 9435 9440 9445 9450 9455 9460 9465 9470 9475 9480 9485 9490 9495 9500 9505 9510 9515 9520 9525 9530 9535 9540 9545 9550 9555 9560 9565 9570 9575 9580 9585 9590 9595 9600 9605 9610 9615 9620 9625 9630 9635 9640 9645 9650 9655 9660 9665 9670 9675 9680 9685 9690 9695 9700 9705 9710 9715 9720 9725 9730 9735 9740 9745 9750 9755 9760 9765 9770 9775 9780 9785 9790 9795 9800 9805 9810 9815 9820 9825 9830 9835 9840 9845 9850 9855 9860 9865 9870 9875 9880 9885 9890 9895 9900 9905 9910 9915 9920 9925 9930 9935 9940 9945 9950 9955 9960 9965 9970 9975 9980 9985 9990 9995 9999

65 70 75 80 85 90 95 100 105 110 115 120 125 130 135 140 145 150 155 160 165 170 175 180 185 190 195 200 205 210 215 220 225 230 235 240 245 250 255 260 265 270 275 280 285 290 295 300 305 310 315 320 325 330 335 340 345 350 355 360 365 370 375 380 385 390 395 400 405 410 415 420 425 430 435 440 445 450 455 460 465 470 475 480 485 490 495 500 505 510 515 520 525 530 535 540 545 550 555 560 565 570 575 580 585 590 595 600 605 610 615 620 625 630 635 640 645 650 655 660 665 670 675 680 685 690 695 700 705 710 715 720 725 730 735 740 745 750 755 760 765 770 775 780 785 790 795 800 805 810 815 820 825 830 835 840 845 850 855 860 865 870 875 880 885 8

Verbindungsnetzwerk auf Verdrahtungsbasis. Nimmt man nun Bezug auf Fig. 16, so ist ein Blockdiagramm einer beispielhaften Ausführungsform einer Allzweck-Verbindungsmatrix 16 gezeigt. In der Fig. 16 handelt es sich bei der GPIM 16 um eine toroidale Verbindungsmausche oder äquivalent um ein "k-ary-2-cube" bzw. ein k-2-Kubus mit einer Anzahl von ersten Kommunikationskanälen 380 und einer Anzahl von zweiten Kommunikationskanälen 382. Jeder erste Kommunikationskanal 380 beinhaltet eine Anzahl von Knotenverbindungssitzen 384, wie dies auch bei jedem zweiten Kommunikationskanal 382 der Fall ist. Jede Verbindungs-I/O-Einheit 304 in dem System 10 ist vorzugsweise mit der GPIM 16 derartig verbunden, daß die Nachrichteneingangsleitung 314 und die Nachrichtenausgangsleitung 316 an fortlaufende Knotenverbindungssitze 384 innerhalb eines gegebenen Kommunikationskanals 380, 382 angeschlossen sind. Bei der bevorzugten Ausführungsform beinhaltet jede T-Maschine 14 eine Verbindungs-I/O-Einheit 304, die mit dem ersten Kommunikationskanal 380, und eine Verbindungs-I/O-Einheit 304, die mit dem zweiten Kommunikationskanal 382 in der oben beschriebenen Weise verbunden ist. Die gemeinsame Schnittstellen- und Steuereinheit 302 innerhalb der T-Maschine 14 erleichtert vorzugsweise das Leiten von Informationen zwischen ihrer Verbindungs-I/O-Einheit 304, die mit dem ersten Kommunikationskanal verbunden ist, und ihrer Verbindungs-I/O-Einheit 304, die mit dem zweiten Kommunikationskanal 382 verbunden ist. Somit erleichtert für eine T-Maschine 14 mit einer Verbindungs-I/O-Einheit 304, die an den ersten Kommunikationskanal angeschlossen ist, der als 380c bezeichnet ist, und einer Verbindungs-I/O-Einheit 304, die mit dem zweiten Kommunikationskanal verbunden ist, der als 382c in Fig. 16 bezeichnet ist, diese gemeinsame Schnittstellen- und Steuereinheit 302 der T-Maschine das Leiten vom Informationen zwischen diesem Satz von ersten und zweiten Kommunikationskanälen 380c, 382c.

Die GPIM 16 erleichtert somit das parallele Leiten von mehreren Nachrichten zwischen den Verbindungs-I/O-Einheiten 304. Für die zweidimensionale GPIM 16, die in Fig. 16 gezeigt ist, beinhaltet jede T-Maschine 14 vorzugsweise eine einzige Verbindungs-I/O-Einheit 304 für den ersten Kommunikationskanal 380 und eine einzige Verbindungs-I/O-Einheit 304 für den zweiten Verbindungskanal 382. Fachleute werden erkennen, daß bei einer Ausführungsform, bei der die GPIM 16 eine Dimensionalität aufweist, die größer als 2 ist, die T-Maschine 14 vorzugsweise mehr als zwei Verbindungs-I/O-Einheiten 304 aufweist. Vorzugsweise ist die GPIM 16 als ein k-ary-2-cube bzw. k-2-Kubus mit einer 16-Bit-Datenpfadgröße realisiert.

Bei der vorhergehenden Beschreibung wurden verschiedene Elemente der vorliegenden Erfindung vorzugsweise realisiert, indem rekonfigurierbare Hardware-Systemelemente verwendet wurden. Die Hersteller von reprogrammierbaren Logikvorrichtungen liefern typischerweise veröffentlichte Richtlinien, um herkömmliche digitale Hardware zu realisieren, indem reprogrammierbare oder rekonfigurierbar Hardware-Systemelemente verwendet werden. Zum Beispiel beinhaltet das 1994 Xilinx Programmable Logic Data Book (Xilinx, Inc., San Jose, GA) als Bemerkungen zur Anwendung bzw. als "Application Notes" das folgende: Application Note XAPP 005.002 "Register-Based FIFO"; Application Note XAPP 044.00 "High-Performance RAM-Based FIFO"; Application Note XAPP 013.001 "Using the Dedicated Carry Logic in the XC 4000"; Application Note XAPP 018.000 "Estimating the Performance of XC 4000 Adders and Counters"; Application Note XAPP 028.001, "Frequency/Phase Comparator for Phase-Locked Loops"; Application Note XAPP 031.000 "Using the XC4000 RAM Capability"; Application Note XAPP 036.001, "Four-Port DRAM Controller..."; und Application Note XAPP 039.001, "18-Bit Pipelined Accumulator". Zusätzliches von Xilinx veröffentlichtes Material beinhaltet Merkmale aus "XCELL, The Quarterly Journal for Xilinx Programmable Logic Users". Zum Beispiel ist ein Artikel in der Ausgabe 14 des dritten Quartals 1994 erschienen, der die Realisierung von schnellen Integermultiplikatoren detailliert beschreibt.

Bei dem hierin beschriebenen System 10 handelt es sich um eine skalierbare, parallele Computer-Architektur, um dynamisch mehrere ISAs zu realisieren. Jede einzelne S-Maschine 12 ist in der Lage, ein gesamtes Computerprogramm selbst ablaufen zu lassen, und zwar unabhängig von einer anderen S-Maschine 12 oder externen Hardware-Systemelementen, wie z. B. einem Hostcomputer. Auf jeder individuellen S-Maschine 12 sind mehrere ISAs sequentiell in der Zeit während der Programmausführung realisiert, und zwar in Antwort auf die Rekonfigurations-Interrupts und/oder auf im Programm eingebettete Rekonfigurationsanweisungen. Weil das System 10 vorzugsweise mehrere S-Maschinen 12 beinhaltet, werden mehrere Programme vorzugsweise simultan ausgeführt, wo jedes Programm unabhängig sein kann. Da somit das System 10 vorzugsweise mehrere S-Maschinen 12 beinhaltet, sind mehrere ISAs simultan (d. h. parallel) zu allen Zeiten, außer während der Systeminitialisierung oder Rekonfigurierung, realisiert. Das heißt, zu jeder gegebenen Zeit werden mehrere Sätze von Programminstruktionen simultan ausgeführt, und zwar wo jeder Satz von Programminstruktionen gemäß einer entsprechenden ISA ausgeführt wird. Jede derartige ISA kann einzig sein.

S-Maschinen 12 kommunizieren miteinander und mit I/O-Vorrichtungen 20 über den Satz von T-Maschinen 14, der GPIM 16 und jeder I/O-T-Maschine 18. Obwohl jede S-Maschine 12 einen ganzen Computer in sich selbst darstellt, der zu einem unabhängigen Betrieb in der Lage ist, ist jede S-Maschine 12 in der Lage als eine Master-S-Maschine 12 für andere S-Maschinen 12 oder das gesamte System 10 zu wirken, Daten zu senden und/oder Befehle zu anderen S-Maschinen 12, einer oder mehreren T-Maschinen 16 oder einer oder mehreren I/O-T-Maschinen 18 oder einer oder mehreren T/O-Vorrichtungen 22 zu senden.

Das System 10 der vorliegenden Erfindung ist somit besonders nützlich für Probleme, die sowohl räumlich als auch zeitlich in ein oder mehrere datenparallele Subprobleme unterteilt werden können, z. B.: Bildverarbeitung, medizinische Datenverarbeitung, kalibriertes Farbmatching bzw. kalibrierte Farbanpassung, Datenbasisrechnen, Vorlagenverarbeitung, assoziative Suchmaschinen und Netzwerkserver. Bezuglich Rechenproblemen mit einem großen Feld von Operanden existiert ein Datenparallelismus, wenn Algorithmen verwendet werden können, um so eine wirksame Berechnungsgeschwindigkeit anzubieten, die durch parallele Rechentechniken beschleunigt wird. Datenparallelprobleme besitzen eine bekannte Komplexität, nämlich von $O(n^k)$. Der Wert von k ist problemabhängig; z. B. $k = 2$ für Bildverarbeitung und $k = 3$ für medizinische Datenverarbeitung. Bei der vorliegenden Erfindung werden individuelle S-Maschinen 12 vorzugsweise verwendet, um den Datenparal-

lelismus bei der Ebene von Programminstruktionen auszuschöpfen bzw. auszunutzen. Weil das System 10 mehrere S-Maschinen 12 beinhaltet, wird das System 10 vorzugsweise verwendet, um den Datenparallelismus auf der Ebene von Sätzen von gesamten Programmen auszunutzen.

Das System 10 der vorliegenden Erfindung liefert eine große Menge an Rechenleistung, und zwar wegen seiner Fähigkeit, die Instruktionenverarbeitungshardware in jeder S-Maschine 12 vollständig zu rekonfigurieren, um die Rechenfähigkeiten einer derartigen Hardware relativ zu Rechnerfordernissen zu einem gegebenen Augenblick zu optimieren. Jede S-Maschine 12 kann unabhängig von jeder anderen S-Maschine 12 rekonfiguriert werden. Das System 10 behandelt vorteilhafterweise jeden Konfigurationsdatensatz und folglich jede ISA als eine programmierte Grenze oder Schnittstelle zwischen Software und der rekonfigurierbaren Hardware, die hierin beschrieben ist. Die Architektur der vorliegenden Erfindung erleichtert zusätzlich die Strukturierung auf hoher Ebene von rekonfigurierbarer Hardware, um selektiv die Belange aktueller Systeme in situ anzusprechen, einschließlich: der Arten und Weisen, in denen eine Unterbrechung die Instruktionenverarbeitung beeinträchtigt; des Erfordernisses einer deterministischen bzw. bestimmbaren Verzögerungsantwort bzw. Latenzantwort, um Echtzeitverarbeitung und Steuerfähigkeiten zu erleichtern; und des Erfordernisses nach wählbaren Antworten zur Fehlerhandhabung.

Im Gegensatz zu anderen Computer-Architekturen lehrt die vorliegende Erfindung die maximale Verwendung von Silizium-Systemelementen zu allen Zeiten. Die vorliegende Erfindung liefert ein paralleles Computersystem, das auf jede gewünschte Größe zu jeder Zeit vergrößert werden kann, sogar zu massiv parallelen Größen mit tausenden von S-Maschinen 12. Derartige Skalierbarkeit bezüglich der Architektur ist möglich, da Instruktionenverarbeitung auf S-Maschinen-Basis absichtlich von der Datenkommunikation auf T-Maschinen-Basis getrennt ist. Dieses Trennungsparadigma bzw. Trennungsbeispiel bezüglich der Befehlsverarbeitung/Datenkommunikation paßt äußerst gut für datenparalleles Rechnen. Die interne Struktur der S-Maschinen-Hardware ist vorzugsweise für den Zeitfluß von Instruktionen optimiert, während die interne Struktur der T-Maschinen-Hardware vorzugsweise für wirksame Datenkommunikation optimiert ist. Der Satz von S-Maschinen 12 und der Satz von T-Maschinen stellen jeweils einen trennbaren, konfigurierbaren Bestandteil in einer Raum-Zeit-Aufteilung einer datenparallelen Rechenarbeit dar.

Mit der vorliegenden Erfindung kann zukünftige rekonfigurierbare Hardware ausgenutzt werden, um Systeme mit sogar noch größeren Rechenfähigkeiten zu konstruieren, während die Gesamtstruktur bzw. die zugrundeliegende Struktur, die hierin beschrieben ist, aufrechterhalten wird. Mit anderen Worten, das System 10 der vorliegenden Erfindung ist technologisch skalierbar. Bei scheinbar allen gegenwärtigen rekonfigurierbaren Logikvorrichtungen handelt es sich um komplementäre Metalloxidhalbleiter- bzw. "Complementary Metal-Oxide Semiconductor" (CMOS) -Technologie auf Speicherbasis. Fortschritte in der Kapazität bzw. Fähigkeit der Bauelemente folgen den Trends der Halbleiterspeichertechnologie. In zukünftigen Systemen würde ein rekonfigurierbares Logikbauelement, das verwendet wird, um eine S-Maschine 12 zu erstellen bzw. zu konstruieren, eine Aufteilung der internen Hardware-Systemelemente in Übereinstimmung mit den Innenschleifen- und Außenschleifen-ISA-Parameter aufweisen, die hierin beschrieben sind. Größere rekonfigurierbare Logikvorrichtungen bieten einfach die Fähigkeit an, mehr datenparallele Rechenarbeit innerhalb eines einzigen Bauelements durchzuführen. Zum Beispiel würde eine größere Funktionseinheit 194 innerhalb der zweiten beispielhaften Ausführung der DOU 63, wie sie oben unter Bezugnahme auf die Fig. 9B beschrieben ist, sich an größere Bildkerngrößen bzw. "imaging kernel sizes" anpassen. Fachleute werden erkennen, daß die technologische Skalierbarkeit, die durch die vorliegende Erfindung bereitgestellt wird, nicht auf Vorrichtungen auf CMOS-Basis beschränkt ist, noch daß sie auf Implementationen auf FPGA-Basis beschränkt ist. Somit liefert die vorliegende Erfindung eine technologische Skalierbarkeit unabhängig von der bestimmten Technologie, die verwendet wird, um eine Rekonfigurierbarkeit oder Reprogrammierbarkeit bereitzustellen.

Nimmt man nun Bezug auf die Fig. 17A und 17B, so ist ein Flußdiagramm eines bevorzugten Verfahrens für skalierbares, paralleles, dynamisch rekonfigurierbares Rechnen gezeigt. Vorzugsweise wird das Verfahren der Fig. 17A und 17B innerhalb jeder S-Maschine 12 in dem System 10 durchgeführt. Das bevorzugte Verfahren beginnt im Schritt 1000 in der Fig. 17A mit der Rekonfigurationslogik 104, die einen Konfigurationsdatensatz, der einer ISA entspricht, ausliest bzw. wieder aufnimmt. Danach konfiguriert im Schritt 1002 die Rekonfigurationslogik 104 jedes Element innerhalb der IFU 60, der DOU 62 und der AOU 64 gemäß dem ausgelesenen Datensatz im Schritt 1002, wodurch eine DRPU-Hardware-Organisation für die Implementation der ISA, die gegenwärtig betrachtet wird, erzeugt wird. Nachfolgend zum Schritt 1002 liest die Interruptlogik 106 die Interruptantwortsignale aus, die in dem Architekturbeschreibungsspeicher 101 gespeichert sind, und erzeugt einen entsprechenden Satz von Übergangssteuersignalen, die festlegen, wie die gegenwärtige DRPU-Konfiguration auf die Unterbrechungen bzw. Interrupts im Schritt 1004 antwortet. Die ISS 100 initialisiert nachfolgend eine Programmzustandsinformation im Schritt 1006, nach der die ISS 100 einen Instruktionausführungszyklus im Schritt 1008 initialisiert.

Danach bestimmt im Schritt 1010 die ISS 100 oder die Interruptlogik 106, ob eine Rekonfiguration erforderlich ist. Die ISS 100 bestimmt, daß die Rekonfiguration in dem Fall erforderlich ist, daß eine Rekonfigurationsanweisung während der Programmausführung gewählt wird. Die Interruptlogik 106 bestimmt, daß eine Rekonfiguration in Antwort auf einen Rekonfigurationsinterrupt erforderlich ist. Falls eine Rekonfiguration erforderlich ist, schreitet das bevorzugte Verfahren zum Schritt 1012 fort, in dem eine Rekonfigurationshandhabungseinrichtung Programmzustandsinformationen sichert. Vorzugsweise beinhalten die Programmzustandsinformationen einen Bezug zu dem Konfigurationsdatensatz, der die laufende DRPU-Konfiguration entspricht. Nach dem Schritt 1012 kehrt das bevorzugte Verfahren zum Schritt 1000 zurück, um einen nächsten Konfigurationsdatensatz derartig auszulesen, wie durch die Rekonfigurationsanweisung oder den Rekonfigurationsinterrupt verwiesen wurde.

Für den Fall, daß eine Rekonfiguration im Schritt 1010 nicht erforderlich ist, bestimmt die Interruptlogik 106,

ob ein Nicht-Rekonfigurationsinterrupt eine Behandlung im Schritt 1014 erfordert. Falls dem so ist, bestimmt die ISS 100 als nächstes im Schritt 1020, ob ein Zustandsübergang von dem gegenwärtigen ISS-Zustand innerhalb des Instruktionsausführungszyklus zu dem Interrupt Servicezustand, basierend auf den Übergangssteuersignalen, möglich ist. Falls ein Zustandsübergang zu dem Interruptservicezustand nicht möglich ist bzw. nicht erlaubt ist, schreitet die ISS 100 zu einem nächsten Zustand in dem Instruktionsausführungszyklus fort und kehrt zu dem Zustand 1020 zurück. Für den Fall, daß die Übergangssteuersignale einen Zustandsübergang von dem gegenwärtigen ISS-Zustand innerhalb des Instruktionsausführungszyklus zu dem Interruptservicezustand erlauben, schreitet die ISS 100 als nächstes zu dem Interruptservicezustand im Schritt 1024 fort. Im Schritt 1024 sichert die ISS 100 die Programmzustandsinformation und führt Programminstruktionen zur Abarbeitung des Interrupts aus. Nachfolgend zum Schritt 1024 kehrt das bevorzugte Verfahren zum Schritt 1008 zurück, um den gegenwärtigen Instruktionsausführungszyklus wieder aufzunehmen, falls er nicht vollendet worden ist, oder um einen nächsten Instruktionsausführungszyklus auszulösen bzw. zu initialisieren.

Für den Fall, daß kein Nicht-Rekonfigurationsinterrupt eine Abarbeitung im Schritt 1014 erfordert, schreitet das bevorzugte Verfahren zum Schritt 1016 fort und bestimmt, ob die Ausführung des gegenwärtigen Programms vollendet ist. Falls die Ausführung des gegenwärtigen Programms fortgesetzt werden soll, kehrt das bevorzugte Verfahren zum Schritt 1008 zurück, um einen anderen Instruktionsausführungszyklus auszulösen bzw. zu initialisieren. Ansonsten endet das bevorzugte Verfahren.

Die Lehren der vorliegenden Erfindung unterscheiden sich entscheidend von anderen Systemen und Verfahren zum reprogrammierbaren oder rekonfigurierbaren Rechnen. Insbesondere ist die vorliegende Erfindung nicht äquivalent zu herunterladbaren Mikrocode-Architekturen, weil derartige Architekturen sich im allgemeinen auf nicht-rekonfigurierbare Steuereinrichtungen und nicht-rekonfigurierbare Hardware verlassen. Die vorliegende Erfindung unterscheidet sich also eindeutig von einem zugeordneten rekonfigurierbaren Prozessor bzw. "Attached Reconfigurable Processor" (ARP)-System, in dem ein Satz von rekonfigurierbaren Hardware-Systemelementen zu einem nicht-rekonfigurierbaren Hostprozessor oder Hostsystem verbunden wird. Ein ARP-Apparat hängt bezüglich der Ausführung einiger Programminstruktionen von dem Host ab. Deshalb ist der Satz an verfügbaren Silizium-Systemelementen nicht maximal über den Zeitrahmen der Programmausführung ausgenutzt, da die Silizium-Systemelemente auf dem ARP-Apparat bzw. dem Host untätig sein werden oder ineffizient genutzt werden, wenn der Host- bzw. der ARP-Apparat mit Daten arbeitet. Im Gegensatz dazu handelt es sich bei jeder S-Maschine 12 um einen unabhängigen Computer, in dem gesamte Programme leicht ausgeführt werden können. Mehrere S-Maschinen 12 führen vorzugsweise Programme simultan aus. Die vorliegende Erfindung lehrt deshalb das maximale Ausnutzen von Silizium-Systemelementen bzw. Silizium-Ressourcen zu allen Zeiten, sowohl für einzelne Programm, die auf einzelnen S-Maschinen 12 ausgeführt werden, als auch für mehrere Programme, die auf dem gesamten System 10 ausgeführt werden.

Ein ARP-Apparat liefert eine Rechen-Beschleunigungseinrichtung für einen bestimmten Algorithmus zu einer bestimmten Zeit und ist als ein Satz von Gattern realisiert, die optimal bezüglich dieses bestimmten Algorithmus verbunden sind. Die Verwendung rekonfigurierbarer Hardware-Systemelemente für Allzweck-Operationen, wie z. B. die Handhabung einer Befehlausführung wird in ARP-Systemen vermieden. Darüber hinaus behandelt ein ARP-System nicht einen gegebenen Satz von untereinander verbundenen Gattern als eine leicht wiederverwendbare Resource bzw. als ein leicht wiederverwendbares Systemelement. Im Gegensatz dazu lehrt die vorliegende Erfindung eine dynamisch rekonfigurierbare Verarbeitungseinrichtung, die für die effiziente Handhabung einer Instruktionsausführung konfiguriert ist, und zwar gemäß einem Instruktionsausführungsmodell, das am besten an die Rechenerfordernisse zu einem bestimmten Moment angepaßt ist. Jede S-Maschine 12 beinhaltet eine Vielzahl von leicht wiederverwendbaren Systemelementen, z. B. die ISS 100, die Interruptlogik 106 und die Abspeicher-/Ausrichtlogik 152. Die vorliegende Erfindung lehrt die Verwendung von rekonfigurierbaren Logik-Systemelementen auf der Ebene von Gruppen von CLBs, IOBs und rekonfigurierbaren Verbindungen eher als auf der Ebene von untereinander verbundenen Gattern. Die vorliegende Erfindung lehrt somit die Verwendung rekonfigurierbarer Logikdesign-Konstruktionen höheren Niveaus, die nützlich bei der Durchführung von Operationen auf einer gesamten Klasse von Rechenproblemen sind, eher als daß sie ein einziges nützliches Gatterverbindungsschema lehrt, das für einen einzigen Algorithmus nützlich ist.

Im allgemeinen sind ARP-Systeme auf die Übersetzung eines bestimmten Algorithmus in einen Satz von untereinander verbundenen Gattern gerichtet. Einige ARP-Systeme streben an, Instruktionen hohen Niveaus in eine optimale Hardware-Konfiguration auf Gatterniveau zu kompilieren bzw. zu übersetzen, was im allgemeinen ein "NP-Hard"- bzw. NP-hart-Problem ist. Im Gegensatz dazu lehrt die vorliegende Erfindung die Verwendung eines Kompilers für dynamisch rekonfigurierbares Rechnen, der Programminstruktionen hohen Niveaus in Assemblersprachinstruktionen gemäß einer variablen ISA in einer sehr geradlinigen Art und Weise kompiliert.

Ein ARP-Apparat ist im allgemeinen nicht in der Lage, sein eigenes Hostprogramm als Daten zu behandeln oder sich selbst zu kontextualisieren bzw. vom Kontext abhängig zu behandeln. Im Gegensatz dazu kann jede S-Maschine in dem System 10 ihr eigenes Programm als Daten behandeln und sich somit leicht selbst kontextualisieren bzw. vom Kontext abhängig behandeln. Das System 10 kann sich leicht selbst durch das Ausführen seiner eigenen Programme simulieren. Die vorliegende Erfindung hat zusätzlich die Fähigkeit, seinen eigenen Kompiler zu kompilieren.

Bei der vorliegenden Erfindung kann ein einziges Programm eine erste Gruppe von Instruktionen beinhalten, die zu einer ersten ISA gehören, eine zweite Gruppe von Instruktionen, die zu einer zweiten ISA gehören, eine dritte Gruppe von Instruktionen, die zu einer noch anderen ISA gehören, usw. Die hierin gelehrt Architektur führt jede derartige Gruppe von Instruktionen aus, indem Hardware verwendet wird, die bezüglich der Ausführungszeit bzw. Laufzeit konfiguriert ist, um die ISA zu realisieren, zu der die Instruktionen gehören. Kein System oder Verfahren nach dem Stand der Technik bietet ähnliche Techniken an.

Die vorliegende Erfindung lehrt weiter ein rekonfigurierbares Interruptschema bzw. Unterbrechungsschema,

bei welche die Interruptverzögerungszeit bzw. die Interruptlatenz, die Interruptpräzision und die programmierbare Zustandsübergangsfreigabe sich gemäß der ISA ändern kann, die gegenwärtig betrachtet wird. Keine analogen Techniken wurden in anderen Computersystemen gefunden. Die vorliegende Erfindung lehrt zusätzlich ein Computersystem mit einer rekonfigurierbaren Datenpfadbitbreite, Adressenbitbreite und rekonfigurierbarer Steuerleitungsbreite im Gegensatz zu dem Stand der Technik von Computersystemen.

5 Während die vorliegende Erfindung unter Bezugnahme auf gewisse bevorzugte Ausführungsformen beschrieben wurde, werden Fachleute erkennen, daß verschiedene Änderungen vorgenommen werden können. Veränderungen und Modifikationen der bevorzugten Ausführungsformen werden durch die vorliegende Erfindung bereitgestellt, die nur durch die folgenden Ansprüche beschränkt ist.

10 Zusammenfassend kann man folgendes bemerken:

Die Erfindung betrifft ein System und ein Verfahren zum skalierbaren, parallelen, dynamisch rekonfigurierbaren Rechnen. Ein Satz von S-Maschinen, eine T-Maschine, die zu jeder S-Maschine korrespondierend ist, eine Allzweck-Verbindungsmautrix (GPIM), ein Satz von I/O-T-Maschinen, ein Satz von I/O-Vorrichtungen und eine Master-Zeitbasiseinheit bilden ein System für skalierbares paralleles, dynamisch rekonfigurierbares Rechnen. 15 Jede S-Maschine ist ein dynamisch rekonfigurierbarer Rechner mit einem Speicher, einer ersten lokalen Zeitbasiseinheit und einer dynamisch rekonfigurierbaren Verarbeitungseinheit (DRPU). Die DRPU wird realisiert bzw. implementiert, indem eine reprogrammierbare Logikvorrichtung verwendet wird, die als eine Instruktionsabrufeinheit (IFU), eine Datenoperationseinheit (DOU) und eine Adressenoperationseinheit (AOU) konfiguriert ist, von denen jede selektiv während einer Programmausführung in Antwort auf einen Rekonfigurationsinterrupt oder der Auswahl einer Rekonfigurationsanweisung, die in einen Satz von Programminstruktionen eingebettet 20 ist, rekonfiguriert wird. Jeder Rekonfigurationsinterrupt und jede Rekonfigurationsanweisung nimmt auf einen Satz von Konfigurationsdaten Bezug, der eine DRPU-Hardware-Organisation spezifiziert, die für Realisierung bzw. Implementation einer bestimmten Instruktionssatzarchitektur (ISA) optimiert ist. Die IFU verwaltet Rekonfigurationsoperationen, Instruktionsabru- und Decodieroperationen, Speicherzugriffsoperationen, und sie 25 gibt Steuersignale an die DOU und die AOU aus, um eine Instruktionsausführung zu erleichtern bzw. zu vereinfachen. Die DOU führt Datenberechnungen durch und die AOU führt Adressenberechnungen durch. Bei jeder T-Maschine handelt es sich um eine Datentransfervorrichtung bzw. um eine Datenübertragungsvorrichtung mit einer gemeinsamen Schnittstellen- und Steuereinheit, einer oder mehrerer Verbindungs-I/O-Einheiten und einer zweiten lokalen Zeitbasiseinheit. Bei der GPIM handelt es sich um ein skalierbares Verbindungsnetzwerk, das eine parallele Kommunikation zwischen T-Maschinen erleichtert bzw. vereinfacht. Der Satz von 30 T-Maschinen und die GPIM erleichtert eine parallele Kommunikation zwischen S-Maschinen.

ANHANG A

35 Instruktionssatz 0

Eine Allzweck-Außenschleifen-ISA

40 1.0 Architekturmodell des Programmierers

Dieser Abschnitt stellt die Sicht der ISA0-Architektur aus der Sicht des Programmierers dar, einschließlich Register, Speichermodell, Aufrufkonventionen bzw. "Calling"-Konventionen von Sprachen hohen Niveaus und eines Interruptmodells.

45 1.1. Register

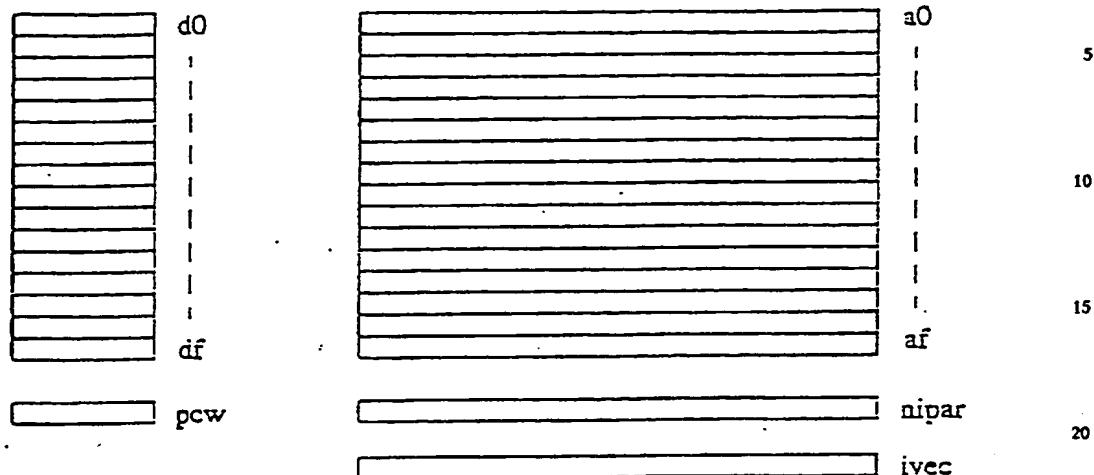
Die ISA0 umfaßt 16 16-Bit-Allzweckregister, 16 Adressenregister, zwei Prozessorstatusregister und ein Interruptvektorregister. Die Mnemoniks für die Daten und Adressenregister verwenden hexadezimale Zahlen, deshalb handelt es sich bei dem letzten Datenregister um df und bei dem letzten Adressenregister um af. Einer der Prozessorstatusregister, nipa (Programmadressenregister für den nächsten Befehl bzw. "Next Instruction Program Address Register"), zeigt zum Abruf auf die Adresse der nächsten Instruktion. Das andere Statusregister pcw (Prozessorsteuerwort bzw. "Processor Control Word") beinhaltet Marken bzw. Flags und Steuerbits, die verwendet werden, um Programmfluß- und Interrupthandhabung zu bewirken. Seine Bits sind in Fig. 2 und 3 festgelegt. Undefinierte Bits sind für eine zukünftige Verwendung reserviert. Die vier Bedingungsflags bzw. Zustandsflags Z, N, V und G werden als Seiteneffekte verschiedener Instruktionen eingestellt bzw. gesetzt. Für eine Zusammenfassung jener Flags bzw. Marken, die durch jede Instruktion beeinträchtigt bzw. beeinflußt werden, siehe Abschnitt 2.0.

60

65

DE 196 14 991 A1

Fig. 1. Register



Das T-(Ablaufverfolgungsmodus bzw. "Trace Mode") und das IM-(Interruptmaske bzw. "Interrupt Mask")-Flag steuert, wie der Prozessor auf die Interrupts antwortet und wann Traps gehandhabt werden. Das Interruptvektorregister ivec hält die 64-Bitadresse der Interruptserviceroutine. Interrupts und Traps werden im Abschnitt 1.4 beschrieben. 25

1.2 Speicherzugriff

Werte, die in dem 64-Bit-Adressenregister gespeichert werden, werden bei Speicher-Lade-/Abspeicherstrukturen verwendet und greifen auf den Speicher in 16- und 64-Bit-Inkrementen zu (siehe Tabelle 4). Bei den Adressen handelt es sich um Bitadressen, d. h. Adresse 16 zeigt auch das Wort, das bei Bit 16 in dem Speicher beginnt. Wörter können nur an 16-Bit-Grenzen gelesen werden und deshalb werden die vier LSBs eines Adressenregisters ignoriert, wenn ein Speicher gelesen wird (siehe [1] zur weiteren Diskussion des Konzepts K1a). 64-Bit-Werte werden als 16-Bit-Worte in der "Little-Endian"-Ordnung (die 16 Bits mit der geringsten Signifikanz werden bei der niedrigsten Adresse gespeichert) gespeichert.

Fig. 2. pcw-Felder

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
T						IMO				L	N	Z	V	C	

Label	Bedeutung
T	Ablaufverfolgungsmodus
IMO	Interrupt-Maskier-Bit
N	Negativ
Z	Null
V	Überlauf
C	Übertrag

1.3. Aufrufkonventionen bzw. "Calling"-Konventionen

Nach Konvention wird ein Register af als der Stackzeiger durch C-Programme verwendet und ein Register ae wird als der Stackrahmenzeiger bzw. "Stack Frame Pointer" verwendet. Die Anemoniks sp und fp können als Alias bzw. als Alternativnamen für diese Register verwendet werden. Alle anderen Register sind zum allgemeinen Gebrauch frei. Der Stack wächst nach unten.

ints sind 16 Bits, longs sind 64, wie dies auch void *s sind. int-Werte werden in d0, long- und void*-Werte in a0 zurückgegeben bzw. returned. d0-d4 und a0-a3 können durch Funktionen "überfahren" bzw. "fertiggemacht" werden, alle anderen allgemeinen Register müssen über die Funktionsaufrufe erhalten bleiben. Nachdem in eine Funktion eingetreten wurde, zeigt der Stackzeiger zu der Rückkehradresse bzw. Returnadresse und somit

beginnt das erste Argument bei Adresse sp + 64 (Dezimal).

1.4. Traps und Interrupts

5 ISA0 arbeitet eine Interruptleitung und Software-Traps von zwei Quellen ab. Alle rufen denselben "Flow-of-Control"-Transfermechanismus bzw. Steuerfluß-Transfermechanismus auf, der unten beschrieben wird.

Extern gibt es einen einzigen INTR-Signaleingang und einen iack-Ausgang. iack wird aktiv, sobald das Interrupt-Maskierbit in pcw gelöscht ist, und zwar indem entweder pcw mit einer xpcw-Instruktion zurückgesetzt wird oder pcw in seinem originalen Wert durch das Rückkehren aus dem Interrupt mit einer rti-Instruktion wiederhergestellt wird. Die Zeitmenge, die zwischen dem Signal geben des Interrupts durch die externe Vorrichtung und dem Abarbeiten des Interrupts durch den Prozessor gebraucht wird, hängt von den Instruktionen ab, die gegenwärtig ausgeführt werden und von der Gegenwart von Software-Traps.

Software-Traps werden entweder durch eine explizite trap-Instruktion oder durch das Ausführen einer Instruktion, bei der das T-(trace)-Flag gesetzt ist, ausgelöst. In diesem Fall wird eine Steuerung zu der Interruptserviceroutine übertragen, und zwar nach der ersten Instruktion, die dem Setzen des T folgt. Wenn eine trap-Instruktion ausgeführt wird, dann setzt der Prozessor das T-Flag und tritt in die Interruptserviceroutine ein als ob das T-Flag vor der Ausführung des Programms gesetzt worden wäre. Es werden keine Interrupts abgearbeitet, während das T-Flag bzw. die T-Marke gesetzt ist. Es werden keine weiteren Traps auftreten, bis das T-Flag entweder durch Rücksetzen von pcw mit einer xpcw-Instruktion oder durch sein Rücksetzen aus dem Stack, indem aus dem Interrupt mit einer rti-Instruktion zurückgekehrt wird, gelöscht wird.

20 Interrupts werden durch die Gegenwart von aktiven Signalen auf dem intr-Externsignal verursacht. Falls das im-Flag oder das T-Flag gesetzt ist, dann werden die Interrupts maskiert und der anhängige Interrupt wird ignoriert. Wenn das im-Flag und das T-Flag gelöscht werden, dann wird die Steuerung zu der Interruptserviceroutine übertragen, und zwar nach der ersten Instruktion, die der Bestätigung von intr folgt. Nach dem Eintritt in die Interruptserviceroutine wird das im-Flag durch den Prozessor gesetzt. Es werden keine weiteren Interrupts auftreten, bis das im-Flag gelöscht ist, und zwar entweder durch ein Rücksetzen von pcw mit einer xpcw-Instruktion oder durch sein Rücksetzen aus dem Stack, indem aus dem Interrupt mit einer rti-Instruktion zurückgekehrt wird.

25 Die Schritte, die von dem Prozessor unternommen werden, wenn ein Interrupt oder ein Trap auftritt, sind die folgenden:

1. Alle Instruktionen, die gegenwärtig ausgeführt werden, werden vollendet.
2. Die Inhalte der 16 Datenregister (d0 zuerst), der 16 Adressenregister (a0 zuerst), pcw, ivec und nipar werden auf den Stack (auf den durch das Register af gezeigt wird) in dieser Ordnung geschoben bzw. gepusht. Der Wert des af, der auf den Stack gepusht wird, ist sein Wert, bevor das Abarbeiten des Interrupts oder des Traps begann.
3. Falls dies ein Interrupt ist, dann wird das Interrupt-Bit in pcw besetzt um weitere Interrupts zu maskieren. Falls dies eine trap-Instruktion ist, dann wird das T-Flag gesetzt. Falls dies ein Trap ist, der durch den T-Flag verursacht wird, dann wird pcw nicht geändert.
4. nipar wird mit dem Wert in dem ivec-Register geladen.
5. Ausführung von Instruktionen in dem Interrupthandler beginnt dann.

Nach der Ausführung der rti-Instruktion werden die folgenden Aktionen unternommen:

1. Die Register werden von dem Stack in der umgekehrten Ordnung, in der sie geschrieben wurden, wiederhergestellt bzw. erneut gespeichert.
2. Die Ausführung wird wieder aufgenommen.

Es ist zu bemerken, daß, falls das Interrupt-Maskierflag nicht bereits gelöst worden ist, es durch die rti-Instruktion gelöscht wird, da es nach dem Eintritt in die Serviceroutine gelöscht bzw. freigegeben war, sofern nicht der Wert des pcw auf dem Stack modifiziert worden ist. Falls das T-Flag durch Ausführen einer trap-Instruktion gesetzt worden ist, dann wird es nach der Ausführung einer rti aus gleichen Gründen gelöscht. Falls der Trap durch das T-Flag verursacht wurde, das vor dem Eintritt in die Serviceroutine gesetzt worden ist, dann muß es durch die Serviceroutine gelöscht werden, um zu bestätigen, daß der Trap aufgetreten ist. Wenn das Interrupt-Maskierflag auf irgendwelche Weise gelöscht worden ist, wird das externe Ausgangssignal iack für einen Taktzyklus aktiv, um der externen Vorrichtung bzw. dem externen Bauelement zu signalisieren, daß der Interrupt abgearbeitet worden ist.

2.0 Instruktionen gruppiert nach Funktion

Die Notationskonventionen sind:

Notation	Bedeutung
d0,d1	Zwei Datenregister (die das selbe Register sein können)
a0,a1	Zwei Adressregister (die das selbe Register sein können)
<<	Links-Schiebe-Operation
>>	Recht-Schiebe-Operation
K _n	N-Bit-Konstante
S K _n	vorzeichenbehaftete (Zweierkomplement) N-bit-Konstante
(d1+l,d1)	Ein 32-Wert von einem Paar von Daten-Register
(a0)	Der Wert bei der Speicherstelle auf die durch das Register a0 gezeigt wird
K _{isa}	Architektur-Konstante, die 4 für ISA0 beträgt, siehe Referenz [1]
AddrWidth	Architektur-Konstante, die die Bit-Breite eines Adressregisters bezeichnet. Ihr Wert ist 64

2.1 Register-Bewegung

Tabelle 1

Register-Bewegung

Mnemonik	Operation	Flags	Seite
mov d0, d1	d1 -> d0	Z, N	17
emov a0, a1	a1 -> a0	Z, N	13
xda d0, a0	a0 - (d0+3, d0+2, d0+1, d0)	KEINE	20
xad a0, d0	(d0+3, d0+2, d0+1, d0) -> a0	KEINE	20
xivec a1	a1 -> ivec	KEINE	21
xpcw	d0 -> pcw	KEINE	21

2.2 Logische Operationen

Tabelle 2

Logische Operationen. Flags modifiziert: Z, N

	Mnemonik	Operation	Flags	Seite
10	and d0, d1	$d1 \leftarrow d1 \wedge d0$	Z, N	10
	or d0, d1	$d1 \leftarrow d1 \vee d0$	Z, N	17
	xor d0, d1	$d1 \leftarrow d1 \oplus d0$	Z, N	20
15	mask d0, d1	$d1 \leftarrow d1 \wedge \neg d0$	Z, N	16
	inv d0, d1	$d1 \leftarrow \neg d0$	Z, N	14
	sl d0, d1	$d1 \leftarrow d1 \ll d0$	Z, N	16
20	rotl d1	$d1 \leftarrow d1 \ll 1$	Z, N, C	17
	ksl K4, d0	$d1 \leftarrow d1 \ll K_4$	Z, N	15
	esl a1	$a1 \leftarrow a1 \ll 1$	Z, N	13
25	erotl a1	$a1 \leftarrow a1 \ll 1 \mid C$	Z, N, C	13
	sr d0, d1	$d1 \leftarrow d1 \gg d0$	Z, N	18
	ksr K4, d1	$d1 \leftarrow d1 \gg K_4$	Z, N	16
30	esr a1	$a1 \leftarrow a1 \gg 1$	Z, N	14
	byte a0, d1	$d1 \leftarrow (d1 \gg (a0_3 \cdot 8))$ -0xff	Z, N ^a	10

a. Das N-Flag wird durch Bit 7 und nicht Bit 15 gesetzt

35

40

45

50

55

60

65

DE 196 14 991 A1

2.3 Speicher Laden/Abspeichern

Tabelle 3

Laden/Abspeichern. Modifizierte Flags: Z, N

5

Mnemonik	Operation	Seite
st d0, a1	$(a1) \leftarrow d0$	18
str d0, a1	$a1 \leftarrow a1 - (1 << K_{isa})$ $(a1) \leftarrow d0$	18
estr a0, a1	$a1 \leftarrow a1 -$ $(1 << K_{isa}) * (\text{AddrWidth}/K_{isa})$ $(a1) \leftarrow a0$	14
ld a0, d1	$d1 \leftarrow (a0)$	16
ldf a0, d1	$d1 \leftarrow (a0)$ $a1 \leftarrow a1 + (1 << K_{isa})$	15
eldf a0, a1	$a1 \leftarrow (a0)$ $a1 \leftarrow a1 +$ $(1 << K_{isa}) * (\text{AddrWidth}/K_{isa})$	12
ldi K16, d0	$d0 \leftarrow K_{16}$	16
eldi K64, a0	$a0 \leftarrow K_{64}$	13

10

15

20

25

30

35

40

45

50

55

60

65

2.4 Arithmetische Operationen

Tabelle 4

5

Arithmetische Operationen. Modifizierte Flags: Z, N, V, C

	Mnemonik	Operation	Seite
10	add d0, d1	$d1 \leftarrow d1 + d0$	9
	adde d0, d1	$d1 \leftarrow d1 + d0 + C$	9
	addq SK ₃ , d1	$d1 \leftarrow d1 + SK_3$	9
15	sub d0, d1	$d1 \leftarrow d1 - d0$	19
	subc d0, d1	$d1 \leftarrow d1 - d0 - C$	19
	mul d0, d1	$(d1+1, d1) \leftarrow d1 \cdot d0$	17
20	umul d0, d1	$(d1+1, d1) \leftarrow d1 \cdot d0$	20
	div d0, d1	$(d1+1, d1) \leftarrow (d1+1, d1) / d0$	11
	udiv d0, d1	$(d1+1, d1) \leftarrow (d1+1, d1) / d0$	19
25	eadd d0, a1	$a1 \leftarrow a1 + d0$	11
	efadd a0, a1	$a1 \leftarrow a1 + a0$	12
	eaddq SK ₃ , a1	$a1 \leftarrow a1 + SK_3$	11
30	esub d0, a1	$a1 \leftarrow a1 - d0$	14
	efsub a0, a1	$a1 \leftarrow a1 - a0$	12
	cmp d0, d1	$\emptyset \leftarrow d1 - d0$	11
35	ecmp a0, a1	$\emptyset \leftarrow a1 + a0$	12

40

45

50

55

60

65

2.5 Steuer-Fluß

Tabelle 5

Steuer-Fluß. Modifizierte Flags: Keine

5

Mnemonik	Operation	Seite
jmp addr ₅₄	nipar \leftarrow addr ₅₄	15
jcc addr ₅₄	(CC==1)? nipar \leftarrow addr ₅₄ : nipar \leftarrow nipar+(1<<K _{isa})	15
brc CC, SK ₁₆	(CC==1)? nipar \leftarrow nipar + SK ₁₆ nipar \leftarrow nipar+(1<<K _{isa})	10
jsr a0, a1	a1 \leftarrow a1 - (1<<K _{isa}) * (AddrWidth/K _{isa}); (a1) \leftarrow nipar nipar \leftarrow a0	15
--- a1	nipar \leftarrow (a1); a1 \leftarrow a1 + (1<<K _{isa}) * (AddrWidth/K _{isa})	18
tu a0	siehe Abschnitt 1.4	19
rti a1	siehe Abschnitt 1.4	17

3.0 Alphabetische Referenz

35

Der Instruktions-Satz für ISA0 ist unten in alphabetischer Ordnung aufgelistet. Die Mnemonik ist mit einer kurzen Beschreibung dargestellt. Unterhalb dieser befindet sich die binäre Kodierung der Instruktion. Jede Zeile in der binären Kodierung ist ein 16-Bit-Wort. Die beeinträchtigten Flags sind dann aufgelistet. Falls nicht anders spezifiziert bzw. beschrieben, werden die Flags gesetzt, indem die Daten verwendet werden, die in dem Bestimmungsregister gespeichert sind. Es wird angenommen, daß nipar bereits zu Beginn der Instruktionsausführung inkrementiert worden ist. Schließlich wird eine Textbeschreibung der Instruktions-Semantik bereitgestellt.

40

Die Notations-Konventionen, die in den binären Kodierungen verwendet werden, werden in der folgenden Tabelle beschrieben. Die Bedingung-Codes sind in Tabelle 7 definiert.

45

50

55

60

65

Tabelle 6
Notations-Konventionen

Notation	Bedeutung
$dddd_a$	Bestimmungs-Datenregister
$dddd_s$	Quellen-Datenregister
$aaaa_d$	Bestimmungs-Adressregister
$aaaa_s$	Quellen-Adressregister
CCCC	Bedingungs-Code
KKKK	nicht-vorzeichenbehaftete 4-Bit-Konstante
SKKKKKKK	vorzeichenbehaftete 8-Bit-Konstante
SKKKKKKKKKKKKKKK	vorzeichenbehaftete 16-Bit-Konstante
KKKKKKKKKKKKKKKK	nicht-vorzeichenbehaftete 16-Bit-Konstante

add (Add data registers)-Addiert Datenregister

1110	$dddd_a$	0000	$dddd_s$
------	----------	------	----------

Flags: Z, N, V, C

Addiert zwei Datenregister, lässt das Ergebnis im Bestimmungsregister.

addc (Add data registers with carry)-Addiert Datenregister mit Übertrag

1110	$dddd_a$	0010	$dddd_s$
------	----------	------	----------

Flags: Z, N, V, C

Addiert zwei Datenregister plus das Carry-Flag bzw. Übertrags-Flag, lässt das Ergebnis im Bestimmungsregister.

addq (Add quick constant)-Addiere schnell Konstante

1100	$dddd_a$	SKKKKKKK	
------	----------	----------	--

Flags: Z, N, V, C

Addiert ein 8-Bit vorzeichenbehaftete (Zweierkomplement) Konstante zu einem Datenregister, lässt das Ergebnis im Register.

DE 196 14 991 A1

and (Bitwise and) — Bitweises und

1111	dd ₄	1000	dd ₄
------	-----------------	------	-----------------

Flags: Z, N

5

Führt das bitweise UND von zwei Datenregister durch, lässt das Ergebnis im Bestimmungsregister.

10

brCC (Conditional branch) — bedingte Verzweigung

0000	0000	0001	CCCC
SKKKKKKKKKKKKKKKKK			

Flags: keine

15

Falls die Bedingung war ist, dann wird (offset $\ll K_{16}$) zu nipar addiert.

20

bru (Unconditional branch) — unbedingte Verzweigung

0000	0000	0001	0000
SKKKKKKKKKKKKKKK			

Flags: keine

25

(offset $\ll K_{16}$) wird zu nipar addiert.

30

byte (Byte align)-Byte-Ausrichtung

1101	dd ₄	1111	aa ₄
------	-----------------	------	-----------------

Flags: Z, N

40

Verschiebe bedingt 8 Bit nach rechts und maskiere. Wird nach einer Lade-Instruktion verwendet, um 8-Bit-Datenworte auszulesen, die von Wort-Offsets gelesen werden. Falls die Adresse, die in dem Quellen-Adressregister enthalten ist, an einer 8-Bit-Grenze (hat Bit 2 gesetzt) liegt, dann wird der Wert des Datenregisters nach rechts um 8 Bit verschoben. Falls die Adresse nicht an einer 8-Bit-Grenze liegt, werden die oberen 8-Bits des Registers gelöscht.

Merke: Der negative Flag wird mit Bit 7 gesetzt, nicht Bit 15. Dies erleichtert die Vorzeichen-Ausdehnung von 8-Bit-Größen.

45

50

cmp (Compare data registers) — Vergleiche Datenregister

55

1110	dd ₄	1000	dd ₄
------	-----------------	------	-----------------

Flags: Z, N, V, C

60

Setzt Flags zum Größenvergleich von zwei Datenregister, indem das Quellenregister von dem Bestimmungsregister substrahiert wird, beeinflusst nur die Flags.

65

DE 196 14 991 A1

div (Signed 32 by 16 division) — Vorzeichenbehaftete 32-durch-16-Division

5	1110	dddd ₄	0101	dddd ₂
---	------	-------------------	------	-------------------

Flags: Z, N, V, C

10 Vorzeichenbehaftete Division einer vorzeichenbehafteten 32-Bit-Integer durch eine vorzeichenbehaftete 16-Bit Integer, gibt den vorzeichenbehafteten 16-Bit Quotienten und den Rest zurück. Der 32-Bit-Dividend wird gespeichert ("little-endian"), und zwar in zwei aufeinanderfolgenden Registern, die von dem Index des Bestimmungsregister ausgehend beginnen. Der 16-Bit-Divisor ist im Quellenregister bzw. Source-Register. Der Rest wird in dem Bestimmungsregister zurückgegeben, und der Quotient wird in dem Register nach dem Bestimmungsregister zurückgegeben (modulo 16). Ein Überlauf tritt auf, falls der Quotient mehr als 16-Bits zur Darstellung benötigt.

eadd (Add data register to address register) — Addiere Datenregister zu Adressregister

20	0101	aaaa ₄	0010	dddd ₂
----	------	-------------------	------	-------------------

Flags: Z, N, V, C

25 Addiert ein Datenregister zu einem Adressenregister, lässt das Ergebnis im Adressenregister.

30 eaddq (Add quick constant to address register) — Addiere schnell Konstante zu Adressenregister

35	1000	aaaa ₄	SKKKKKKK	
----	------	-------------------	----------	--

35 Flags: Z, N, V, C

Addiert eine vorzeichenbehaftete 8-Bit-Konstante zu einem Adressenregister, lässt das Ergebnis in dem Adressenregister.

40 ecmp (Compare of address registers) — Vergleich von Adressregister

45	0101	aaaa ₄	0100	aaaa ₂
----	------	-------------------	------	-------------------

Flags: Z, N, V, C

50 Setzt Flags zum Größenvergleich von zwei Adressregister, indem das Quellenregister von dem Bestimmungsregister substrahiert wird, beeinflusst nur die Flags.

efadd (Add address register to address register) — Addiere Adressregister zu Adressregister

55	0101	aaaa ₄	0110	aaaa ₂
----	------	-------------------	------	-------------------

Flags: Z, N, V, C

60 Addiert zwei Adressregister, lässt das Ergebnis im Bestimmungsregister.

65

DE 196 14 991 A1

efsub (Subtract address register from address register) — Substrahiere Adressregister von Adressregister

0101	aaaa _d	0111	aaaa _s
------	-------------------	------	-------------------

Flags: Z, N, V, C

5

Substrahiere das Quellenregister von dem Bestimmungsregister, speicher das Ergebnis in dem Bestimmungsregister ab.

10

eldf (Extended load forward) — Erweitertes Laden vorwärts

0110	aaaa _d	1101	aaaa _s
------	-------------------	------	-------------------

Flags: Z, N

15

Post-Inkrement Laden in das Adressregister. Speicher wird von der Adresse gelesen, auf die durch das Quellenregister gezeigt wird, und wird in dem Bestimmungsregister plaziert. Das Quellenregister wird dann inkrementiert.

20

eldi (load immediate into adress register) — Lade sofort in Adressregister

25

1110	aaaa _d	1000	0000
KKKKKKKKKKKKKKKKKK			

Flags: Z, N

30

35

40

Lade 64-Bit-Konstante in ein Adressregister.

emov (Move address register) — Bewege Adressenregister

45

0101	aaaa _d	1111	aaaa _s
------	-------------------	------	-------------------

Flags: Z, N

50

Bewege den Wert von dem Quellenregister zu dem Bestimmungs-Adressregister.

erotl (Rotate address register left through carry flag) — Rotiere bzw. verschiebe zyklisch Adressenregister nach links durch Carry-Flag

55

0101	aaaa _d	1010	0000
------	-------------------	------	------

Flags: Z, N, C

60

Schiebe ein Adressregister zu dem linken Bit. Das LSB wird durch den Wert des Carry-Flag ersetzt. Das MSB wird in das Carry-Flag am Ende der Instruktion plaziert.

65

DE 196 14 991 A1

esl (Shift address register left) — Schiebe Adressenregister links

5	0101	aaaa ₄	1000	0000
Flags: Z, N				

10 Schiebe ein Adressenregister zu dem linken Bit.

10 esr (shift address register right) — Schiebe Adressenregister rechts

15	0101	aaaa ₄	1001	0000
Flags: Z, N				

20 Schiebe ein Adressenregister zu dem rechten Bit.

20 est (Store address register) — Speichere Adressenregister

25	0110	aaaa ₄	1000	aaaa ₄
Flags: Z, N				

30 Speichere aus einem Adressenregister. Der 64-Bit-Wert in dem Quellenregister wird in die Speicherstelle geschrieben, auf die durch das Bestimmungsregister gezeigt wird. Der Wert wird als vier 16-Bit-Wörter, die in der "little-endian"-Ordnung plaziert sind, geschrieben.

35 estr (Extended store reverse) — Erweiterte Speicherumkehrung

35	0110	aaaa ₄	1110	aaaa ₄
Flags: Z, N				

40 Prä-dekrement-speichern aus Adressenregister. Das Bestimmungsregister wird dekrementiert und dann wird der Wert in dem Quellenregister in die Speicherstelle geschrieben, auf die durch das Bestimmungsregister gezeigt wird. Der Wert wird als vier 16-Bit-Wörter geschrieben, die in "little-endian"-Ordnung plaziert sind.

45 esub (Subsfract data register from address register) — Substrahiere Datenregister von Adressregister

50	1110	aaaa ₄	0010	dddd ₄
Flags: Z, N, V, C				

55 Substrahiert ein Datenregister von einem Adressenregister, lässt das Ergebnis im Adressregister.

60 inv (Bitwise inverse) — Bitweise invers

60	1111	dddd ₄	0101	dddd ₄
Flags: Z, N				

65

Plaziert das bitweise Inverse des Quellenregisters in das Bestimmungsregister.

DE 196 14 991 A1

jCC (Conditional jump) — Bedingter Sprung

0000	0000	0000	CCCC
KKKKKKKKKKKKKKKKKK			

Flags: Keine

5

10

15

Bedingter Sprung zu Absolutadressen. Siehe Tabelle 7 für Bedingungscode-Bit-Definitionen.

jmp (Unconditional jump) — Unbedingter Sprung

20

0000	0000	0000	0000
KKKKKKKKKKKKKKKKKK			

Flags: Keine

25

30

35

Unbedingter Sprung zu Absolut-Adresse. Dasselbe wie jCC mit Bedingung "immer".

jsr (Jump to subroutine) — Springe zu Unterprogramm

40

0001	aaaa _d	0000	aaaa _s
------	-------------------	------	-------------------

Flags: Keine

45

Das Bestimmungsregister wird zuerst inkrementiert, dann wird das gegenwärtige nipar (zeigt zu der nächsten Adresse) unter der Adresse abgespeichert, auf die durch das Bestimmungsregister (üblicherweise der Stackpointer) gezeigt wird. nipar wird dann mit der Adresse in dem Quellenregister geladen, bevor die nächste Instruktion geholt bzw. abgerufen wird.

50

ksl (Shift left by constant) — Schiebe links um Konstante

55

1101	dddd _d	1000	KKKK
------	-------------------	------	------

Flags: Z, N

60

Schiebe ein Datenregister um eine konstante Anzahl von Bits nach links.

65

DE 196 14 991 A1

ksr (Shift right by constant) — Schiebe rechts um Konstante

5	1101	dddd _d	1001	KKKK
---	------	-------------------	------	------

Flags: Z, N

10 Schiebe ein Datenregister um eine konstante Anzahl von Bits nach rechts.

Id (Load data register) — Lade Datenregister

15	0110	dddd _d	0001	aaaa _s
----	------	-------------------	------	-------------------

Flags: Z, N

20 Lade ein Datenregister aus dem Speicher. Der Wert, auf den durch das Quellenadreßregister gezeigt wird, wird in das Bestimmungsdatenregister geladen.

Idf (Load forward) — Lade vorwärts

25	0110	dddd _d	0101	aaaa _s
----	------	-------------------	------	-------------------

30 Flags: Z, N

35 Post-inkrement-laden in Datenregister. Speicher wird von der Adresse gelesen, auf die durch das Quellenadressenregister gezeigt wird, und wird in das Bestimmungsdatenregister plaziert. Das Quellenregister wird dann inkrementiert.

Idi (Load immediate) — Lade sofort

40	0111	dddd _d	0000	0000
----	------	-------------------	------	------

KKKKKKKKKKKKKKKKKK

Flags: Z, N

45

Lade einen 16-Bit-Sofort-Wert bzw. einen 16-Bit-Direkt-Wert in ein Datenregister.

mask (Bitwise mask operation)-Bitweise Maskier-Operation

50	1111	dddd _d	0100	dddd _s
----	------	-------------------	------	-------------------

Flags: Z, N

55

Das Bestimmungsregister wird durch das bitweise Inverse des Quellenregisters ersetzt, das mit dem Bestimmungsregister gegründet wird.

60 mov (Move data register) — Bewege Datenregister

65	1111	dddd _d	1010	dddd _s
----	------	-------------------	------	-------------------

Flags: Z, N

Der Wert in dem Quellendatenregister wird in das Bestimmungsdatenregister plaziert.

DE 196 14 991 A1

mul (Signed 16 by 16-bit multiply)-Vorzeichenbehaftete 16 mit 16-Bit-Multiplikation

1110	dd ₄ dd ₃	0100	dd ₄ dd ₃
------	---------------------------------	------	---------------------------------

5

Flags: Z, N, V, C

Das Ergebnis der Multiplikation des Wertes in dem Quellenregister mit dem Wert in dem Bestimmungsregister wird gespeichert ("little-endian"), und zwar in den zwei aufeinanderfolgenden Registern, die mit dem Bestimmungsregister beginnen. 10

or (Bitwise or) — Bitweises oder

15

1111	dd ₄ dd ₃	1110	dd ₄ dd ₃
------	---------------------------------	------	---------------------------------

20

Flags: Z, N

Führt das bitweise ODER bzw. OR von zwei Datenregistern durch, lässt das Ergebnis in dem Bestimmungsregister. 25

rotl (Rotate data register left through carry flag) — Rotiere bzw. verschiebe zyklisch Datenregister links durch Carry-Flag

1101	dd ₄ dd ₃	1010	0000
------	---------------------------------	------	------

30

Flags: Z, N, C

Schiebe ein Datenregister zu dem linken Bit. Das LSB wird durch den Wert des Carry-Flag ersetzt. Das 35 ursprüngliche MSB wird in dem Carry-Flag am Ende der Instruktion plaziert.

rti (Return from interrupt) — Return vom Interrupt

40

0001	aaaa ₄	10001	0000
------	-------------------	-------	------

45

Flags: Z, N, V, C

Siehe Abschnitt 1.4. Das Quellenregister bzw. Sourceregister wird als Stack-Zeiger verwendet.

rts (Return from subroutine) — Rückkehr von Unterprogramm

50

0001	aaaa ₄	0001	0000
------	-------------------	------	------

55

Flags: Keine

Keine Kehre von Unterprogramm zurück, nipar wird aus der Speicherstelle geladen, auf die durch das Bestimmungsregister (üblicherweise der Stack-Zeiger) gezeigt wird. Das Bestimmungsregister wird dann inkrementiert. 60

sl (Shift left) — Schiebe links

65

1101	dd ₄ dd ₃	0000	dd ₄ dd ₃
------	---------------------------------	------	---------------------------------

Flags: Z, N

DE 196 14 991 A1

Das Bestimmungsregister wird links um die Anzahl von Bits verschoben, die durch den Wert des Quellenregister spezifiziert sind.

5 sr (Shift right) — Schiebe rechts

5

1101	dddd ₄	0001	dddd ₄
------	-------------------	------	-------------------

10 Flags: Z, N

Das Bestimmungsregister wird rechts um die Anzahl von Bits verschoben, die durch den Wert des Quellenregister spezifiziert sind.

15

st (Store) — Speichern

20

0110	dddd ₄	0000	aaaa ₄
------	-------------------	------	-------------------

Flags: Z, N

25 Speichere aus einem Datenregister. Der Wert in dem Quellenregister wird zu der Speicherstelle geschrieben, auf die durch das Bestimmungsregister gezeigt wird.

str (Store reserve) — Speicher umgekehrt

30

0110	dddd ₄	0110	aaaa ₄
------	-------------------	------	-------------------

35

Prä-dekrement-speichern aus Datenregister. Das Bestimmungregister wird dekrementiert und dann wird der Wert in dem Quellenregister in die Speicherstelle geschrieben, auf die durch das Bestimmungsregister gezeigt wird.

40

sub (Subtract data register from data register) — Subtrahiere Datenregister von Datenregister

45

1110	dddd ₄	0001	dddd ₄
------	-------------------	------	-------------------

Flags: Z, N, V, C

50 Subtrahiere das Quellenregister von dem Bestimmungsregister, speichere das Ergebnis in dem Bestimmungsregister.

subc (Subtract with carry) — Subtrahiere mit Carry bzw. Übertrag

55

1110	dddd ₄	0011	dddd ₄
------	-------------------	------	-------------------

Flags: Z, N, V, C

60

Subtrahiere das Quellenregister von dem Bestimmungsregister, dann subtrahiere das Carry-Bit, wobei das Ergebnis in dem Bestimmungsregister gespeichert wird.

65

DE 196 14 991 A1

trap (Unconditional trap) — Unbedingter Trap

0011	aaaa ₄	0000	0000
------	-------------------	------	------

5

Flags: Keine

Führe Interrupt-Handler aus. Siehe Abschnitt 1.4. Das Bestimmungsregister wird als der Stack-Pointer verwendet. 10

udiv (Unsigned 32 by 16-bit division) — Nicht-vorzeichenbehaftete 32 durch 16-Bit Division

1110	dddd ₄	0111	dddd ₄
------	-------------------	------	-------------------

15

Flags: Z, N, V, C

20

Nicht-vorzeichenbehaftete Division einer 32-Bit-Integer durch eine vorzeichenbehaftete 16-BitInteger, wobei der vorzeichenbehaftete 16-Bit-Quotient und -Rest zurückgegeben wird. Die 32 Bits werden in zwei aufeinander folgenden Registern, beginnend von dem Index des Bestimmungsregisters, gespeichert ("little endian"). Der Divisor ist in dem Quellenregister. Der Rest wird in das Bestimmungsregister zurückgegeben und der Quotient wird in das nächste Register nach dem Bestimmungsregister zurückgegeben. Ein Überlauf tritt auf; falls der Quotient mehr als 16-Bit zum Darstellen erfordert. 25

umul (Unsigned 16 by 16-Bit-Multiplikation) — Nicht-vorzeichenbehaftete 16 mit 16-Bit Multiplikation

30

1110	dddd ₄	0110	dddd ₄
------	-------------------	------	-------------------

35

Das Ergebnis der Multiplikation des Werts in dem Quellenregister mit dem Wert in dem Bestimmungsregister wird gespeichert ("little-endian"), und zwar in zwei aufeinander folgende Register beginnend mit dem Bestimmungsregister. 40

xad (Transfer address register to data register) — Übertrage Adressenregister zu Datenregister

40

010 1	dddd ₄	1101	aaaa ₄
-------	-------------------	------	-------------------

45

Flags: Keine

50

Übertrage den Wert in dem Quellenadressregister zu vier aufeinander folgende Datenregister beginnend mit dem Bestimmungsregister. Der Wert wird "little-endian"-gespeichert und die Bestimmungsregisteradresse wird modulo 16 berechnet, so daß das Bestimmungsregister jedes Register sein kann.

xda (Transfer data register to address register) — Übertrag Datenregister zu Adressenregister

55

0101	aaaa ₄	1100	dddd ₄
------	-------------------	------	-------------------

60

Übertrag einen "little-endian"-64-Bit-Wert in vier aufeinanderfolgende Datenregister in das Bestimmungsregister. Die Quellenregisteradresse wird modulo 16 berechnet, so daß das Bestimmungsregister jede Integer sein kann. 65

DE 196 14 991 A1

xor (Bitwise exclusive or) — Bitweises exklusives Oder

5	1111	dddd _d	0110	dddd _s
Flags: Z, N				

10 Führt das bitweise exklusive ODER bzw. OR bezüglich zwei Datenregistern durch, lässt das Ergebnis im Bestimmungsregister.

xpcw (Exchange processor control word) — Tausche Prozessor-Steuerwort

15	0100	dddd _d	1010	0000
Flags: Alle				

20 Der Wert in dem Quellendatenregister wird durch das pcw-Register ausgetauscht.

xivec (Exchange interrupt vector) -Tausche Interrupt-Vektor

25	0101	aaaa _d	0111	0000
Flags: Alle				

30 Der Wert in dem Quellenadressregister wird mit dem ivec-Register ausgetauscht.

4.0 Bedingungs-Codes

35 Die Bedingungs-Code-Opcode-Unterfelder verwenden die Werte der folgenden Tabelle:

Tabelle 7

40 Bedingungs-Codes

Mne.	Code	Bedeutung	Gleichung
45 eq	0001	gleich	z
ne	0010	nicht gleich	-z
gt	0011	größer als	$z > (N \oplus V)$
50 lt	0100	kleiner als	$N \oplus V > z$
ge	0101	größer oder gleich	$-(N \oplus V) > z$
le	0110	kleiner oder gleich	$z > (N \oplus V)$
55 p	0111	positiv	-N
n	1000	negativ	N
55 gtu	1001	größer als, nicht vorzeichenbehaftet	$-C > z$
60 vc	1010	Überlauf löschen	-V
vs	1011	Überlauf setzen	V
cc	1100	Übertrag löschen	-C
65 cs	1101	Übertrag setzen	C

Anhang B

Instruktionssatz 1

Eine gepipelinte Multiplikations-Akkumulier-ISA

5

ISA1 gepipelinte Faltungsmaschine für XC 4013

Einführung

10

ISA1 ist ein gepipelinte Multiplikations-Akkumulator-Array, das zu 4 simultanen Multiplikations-Akkumulationen pro Instruktionszyklus in der Lage ist. Es gibt 8 8-Bit Datenregister ($xd0$ – $xd3$ & $yd0$ – $yd3$), einen für jeden Eingang zu den vier 8-Bit-X-8-Bit-Multiplizierern. Die vier Multiplizierer werden über ein gepipelinetes Addier-Array summiert, bis eine engültige 16-Bit-Summe herauskommt, und zwar wo bis zu vier 16-Bit-Register das Ergebnis ($m0$ – $m4$) speichern werden können. Die Architektur der ISA1 vermutet einen Fluß-durch-Batch-Verarbeitungszyklus mit Hauptspeicher. Für sich genommen gibt es keinen Feedback-Pfad bzw. Rückführ-Pfad durch den Multiplikations-Akkumulator-Datenpfad, um akkumulierte Ergebnisse wieder aufzuarbeiten, weil die Betonung auf den Speicher-Datenflüsse liegt. Es ist nichts vorgesehen für Überlauf-Skalierung oder erweiterte Endlichkeits-Akkumulierungen; ISA1 vermutet, daß die Koeffizienten, die für Faltungs-Filtern verwendet werden, nicht mehr als 16-Bit-Ergebnis-Endlichkeiten für alle Datensätze hervorbringen. Das Multiplikations-Array nimmt 8-bit-2er-Komplement-Dateneingänge an und erzeugt ein 16-Bit-2er-Komplement-Ergebnis.

Speicherzugriff wird durch zwei 64-Bit-Addresenregister ($a0$ & $a1$) verwaltet bzw. gehandhabt, die als miteinander vertauschbare Quellen- und Bestimmungsregister gedacht werden können. Programmfluss wird durch das Standard-64-Bit-NIPAR-Register verwaltet und ein 64-Bit-Interrupt-Vektor-Register (IVEC) wird für einen einzelnen Interrupt, wie z. B. einen Frame- oder Datenfertig-Interrupt, unterstützt.

Der Instruktionssatz von ISA1 ist sehr klein und auf 16-Bitwort-Größe ausgerichtet, die mit der $K_{ISA} = 4$ -Speicher-Organisation für den Außenschleifen-Prozessor ISA0 zusammenpaßt. Bis zu 7 Arithmetikoperationen können in einem einzigen Taktzyklus mit ISA1 sofort ausgeführt bzw. beispielhaft durchgeführt werden, und die Implementation hält eine Ergebnisrate von eins pro Takt über ein schmales Fenster von Taktzyklen aufrecht, und zwar mit der Fähigkeit neue Quell- oder Bestimmungsadressen zu indizieren und Registerdaten aus oder zu einem Speicher zu bewegen, und zwar parallel mit einer Berechnung.

ISA1 Instruktionssatz

35

Datenbewegung

ld (reg-vector)

Jeder von bis zu 14 Registern wird sequentiell aus dem Speicher geladen, und zwar gemäß 14-Bit-Bitmap-reg-vector, der rechts-justiert im Instruktionswort enthalten ist.

st (reg-vector)

Jedes von bis zu 14 Register wird sequentiell in einen Speicher gespeichert, und zwar gemäß einem 14-Bit-Bit-map-reg-vector, der rechts-justiert in dem Instruktionswort enthalten ist.

ld (ivec-data)

Die 64-BitAdresse, die dieser Instruktion folgt, wird in das IVEC-Register geladen, während NIPAR + = 5, um auf die nächste auszuführende Instruktion zu zeigen.

Programm-Steuerung

55

jmp (nipar-data)

Die 64-Bit-Adresse, die dieser Instruktion folgt, wird in das NIPAR-Register geladen, wodurch auf die nächste auszuführende Instruktion gezeigt wird.

Arithmetik

60

mac (m-reg)

Das Multiplikations-Ergebnis-Register, auf das durch den 2-Bit-m-reg-Code gezeigt wird, empfängt das Produkt und die Summe $(xd0 \cdot yd0) + (xd1 \cdot yd1) + (xd2 \cdot yd2) + (xd3 \cdot yd3)$.

65

macp (s-vec, d-vec)

Das Multiplikations-Ergebnis-Register, auf das durch 2-Bits des 4-Bit-d-vec-Codes gezeigt wird, empfängt das Produkt und die Summe $(xd0*yd0) + (xd1*yd1) + (xd2*yd2) + (xd3*yd3)$. Ein anderes Bit des d-vec-Codes 5 enables selektiv eine Speicher-Schreiboperation dieses Ergebnis-Registers bei Adresse (a1) bzw. gibt diese frei, während das übrige Bit des d-vec-Codes auswählt, ob das Adressenregister a1 inkrementiert wird oder nicht. Der 8-Bit-s-vec wird in vier 2-Bit-Gruppen aufgeteilt, die sukzessiv für die Datenregister xd0-xd3 spezifizieren bzw. 10 vorgeben, ob eine Leseoperation aus dem Speicher bei Adresse (a0) auftreten soll und ob das Adressenregister a0 inkrementiert werden soll. Falls Lese- oder Schreiboperationen spezifiziert bzw. vorgegeben sind, werden sie parallel mit der Multiplikation durchgeführt. Die Software muß der gepipelineten Ausrichtung der Instruktionsverarbeitung bezüglich Batches bzw. Blöcken von Daten Rechnung tragen, die vom Speicher gelesen werden und im Speicher gespeichert werden.

Rekonfiguration

15

reconf (ISA-vector)

ISA1 wird de-kontextet und die S-Maschine wird für die ISA rekonfiguriert, die durch das ISA-vector-Bitfeld 20 in der Instruktion ausgewählt ist.

20

25

30

35

40

45

50

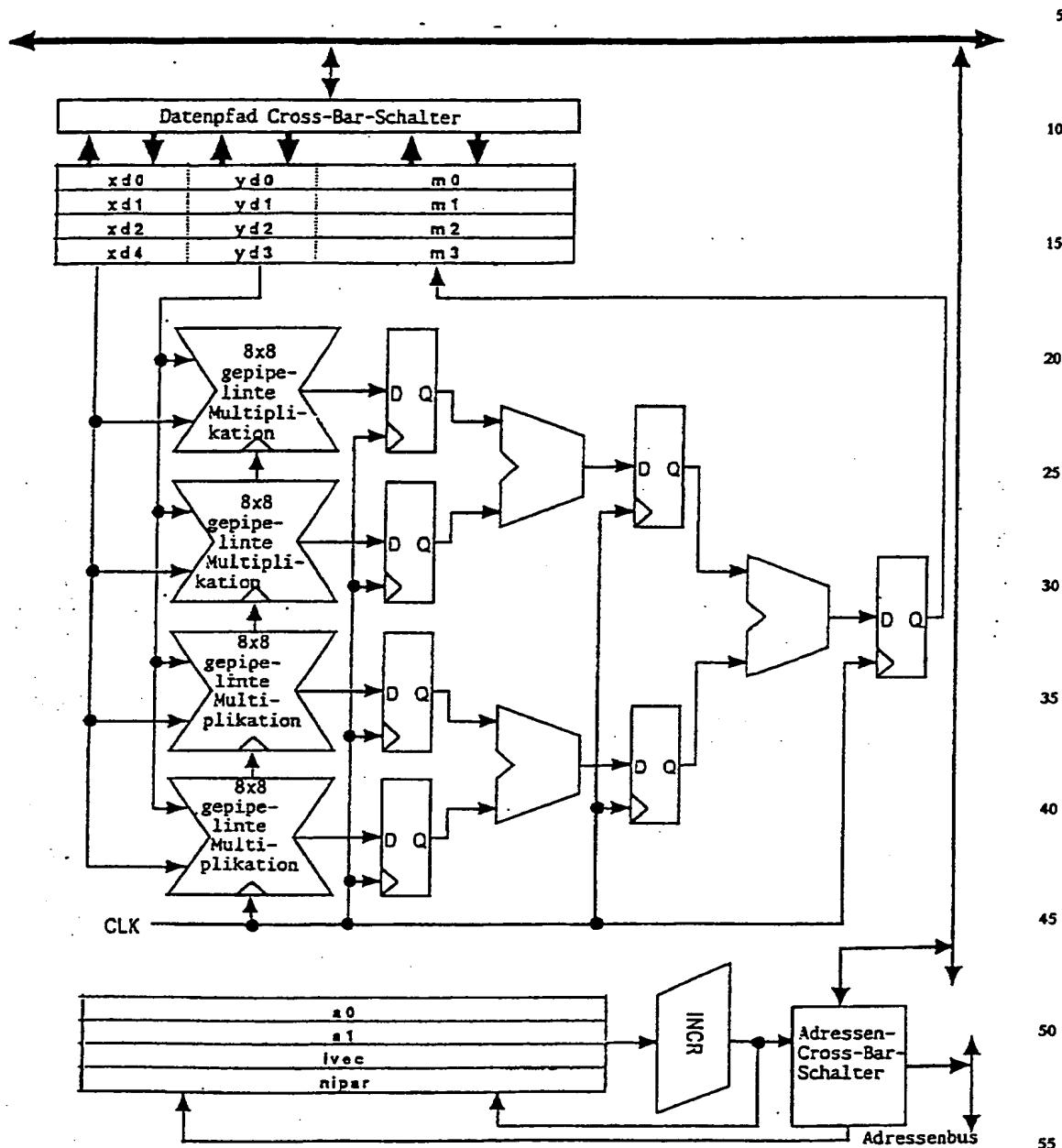
55

60

65

Block-Diagramm von ISA1 — Gepipelnete Faltungsmaschine für XC4013

Speicher-Datenbus



Patentansprüche

1. Dynamisch rekonfigurierbare Verarbeitungseinheit zum Ausführen von Programminstruktionen, um Daten zu verarbeiten, wobei die dynamisch rekonfigurierbare Verarbeitungseinheit einen Eingang, einen Ausgang und eine veränderbare interne Hardware-Organisation umfaßt, die selektiv während der Ausführung einer Sequenz von Programminstruktionen zwischen einer ersten Hardware-Architektur, die Instruktionen von einem ersten Instruktionssatz ausführt, und einer zweiten Hardware-Architektur, die Instruktionen eines zweiten Instruktionssatzes ausführt, veränderbar ist, wobei die dynamisch rekonfigurierbare Verarbeitungseinheit, wenn sie als die erste Hardware-Architektur konfiguriert ist, auf eine Rekonfigurationsinstruktion anspricht, um die interne Hardware-Organisation der dynamisch rekonfigurierbaren Verarbeitungseinheit zu verändern, die als die zweite Hardware-Architektur konfiguriert werden soll.

2. Dynamisch rekonfigurierbare Verarbeitungseinheit nach Anspruch 1, bei welcher die Rekonfigurationsinstruktion eine der Instruktionen in dem ersten Instruktionssatz ist.

3. Dynamisch rekonfigurierbare Verarbeitungseinheit nach Anspruch 1, bei welcher die Rekonfigurationsinstruktion ein Teil einer anderen Instruktion ist und die Ausführung der Rekonfigurationsinstruktion von Daten abhängt, die in Registern der dynamisch rekonfigurierbaren Verarbeitungseinheit gespeichert sind.

4. Dynamisch rekonfigurierbare Verarbeitungseinheit nach Anspruch 1, die weiter folgendes umfaßt: eine zweite rekonfigurierbare Verarbeitungseinheit mit einem Eingang, einem Ausgang und einer veränderbaren internen Hardware-Organisation, die selektiv während der Ausführung einer Folge von Programminstruktionen zwischen der ersten Hardware-Architektur, die Instruktionen von dem ersten Instruktionssatz ausführt, und der zweiten Hardware-Architektur, die Instruktionen von dem zweiten Instruktionssatz ausführt, veränderbar ist, wobei der Eingang der zweiten rekonfigurierbaren Verarbeitungseinheit mit dem Ausgang der dynamisch rekonfigurierbaren Verarbeitungseinheit verbunden ist und der Ausgang der zweiten rekonfigurierbaren Verarbeitungseinheit mit dem Eingang der dynamisch rekonfigurierbaren Verarbeitungseinheit verbunden ist.

5. Dynamisch rekonfigurierbare Verarbeitungseinheit nach Anspruch 4, bei welcher die dynamisch rekonfigurierbare Verarbeitungseinheit unabhängig von einer Rekonfiguration der zweiten rekonfigurierbaren Verarbeitungseinheit dynamisch rekonfigurierbar ist.

6. Dynamisch rekonfigurierbare Verarbeitungseinheit nach Anspruch 1, bei welcher es sich bei der ersten Hardware-Architektur um einen seriellen Instruktionssprozessor bzw. eine serielle Instruktionenverarbeitungseinrichtung handelt und bei der zweiten Hardware-Architektur um einen parallelen Instruktionssprozessor bzw. eine parallele Instruktionenverarbeitungseinrichtung.

7. Dynamisch rekonfigurierbare Verarbeitungseinheit nach Anspruch 1, bei welcher die veränderbare interne Hardware-Organisation der dynamisch rekonfigurierbaren Verarbeitungseinheit eine Instruktionssabrufeinheit mit einem Dateneingang, einem ersten Steuerausgang und einem zweiten Steuerausgang umfaßt, um Instruktionsausführungsoperationen innerhalb der dynamisch rekonfigurierbaren Verarbeitungseinheit sequentiell zu ordnen, wobei der Dateneingang mit einem Datenport eines Speichers verbunden ist.

8. Dynamisch rekonfigurierbare Verarbeitungseinheit nach Anspruch 7, bei welcher die Instruktionssabrufeinheit weiter folgendes aufweist:

30 einen Architekturbeschreibungsspeicher mit einem Ausgang, wobei der Architekturbeschreibungsspeicher einen Satz von Architekturbeschreibungssignalen einschließlich eines Interruptantwortsignals speichert, das eine Art und Weise spezifiziert, in der die dynamisch rekonfigurierbare Verarbeitungseinheit auf ein Interruptsignal antwortet, wenn sie konfiguriert ist, um eine Instruktionssatz-Architektur zu verwirklichen bzw. zu implementieren;

35 eine Instruktionzzustandsfolgesteuereinheit bzw. ein Instruktionzzustands-Sequenzer mit einem Eingang und einem Ausgang, um einen Instruktionsausführungszyklus und einen Übergang zwischen einem Instruktionssabruftzustand, einem Instruktionsedecodierzustand, einem Instruktionsausführungszustand und einem Schreib-zurück-Zustand zu steuern; und

40 eine Interruptzustandsmaschine mit einem Eingang und einem Ausgang, um ein Übergangssteuersignal zu erzeugen, das einen Zustand, innerhalb des Instruktionsausführungszyklus spezifiziert, für den ein Übergang zu einem Interruptservicezustand erlaubt ist, wobei der Eingang der Interruptzustandsmaschine mit dem Ausgang des Architekturbeschreibungsspeichers verbunden ist, der Ausgang der Interruptzustandsmaschine mit dem Eingang der Instruktionzzustands-Folgesteuereinheit bzw. dem Instruktionzzustands-Sequenzer verbunden ist.

45 9. Dynamisch rekonfigurierbare Verarbeitungseinheit nach Anspruch 1, bei welcher die veränderbare interne Hardware-Organisation der dynamisch rekonfigurierbaren Verarbeitungseinheit eine Datenoperationseinheit mit einem Datenport und einem Steuereingang umfaßt, um Operationen bezüglich der Daten durchzuführen, wobei der Datenport der Datenoperationseinheit mit dem Datenport des Speichers verbunden ist und der Steuereingang verbunden ist, um Steuersignale zu empfangen.

50 10. Dynamisch rekonfigurierbare Verarbeitungseinheit nach Anspruch 9, bei welcher die Datenoperationseinheit folgendes umfaßt:

55 einen Schalter mit einem Datenport, einen Steuereingang, einen Rückführ- bzw. Feedbackeingang und einen Ausgang, um selektiv Daten zwischen dem Datenport, dem Rückführ- bzw. Feedbackeingang und dem Ausgang zu leiten, wobei der Datenport des Schalters mit dem Datenport des Speichers verbunden ist und der Steuereingang des Schalters verbunden ist, um Steuersignale zu empfangen;

60 eine Abspeicher-/Ausrichteinheit mit einem Eingang, einem Ausgang und einer Steuereinheit, um Daten zu speichern, wobei der Eingang der Abspeicher-/Ausrichteinheit mit dem Ausgang des Schalters verbunden ist und der Steuereingang der Abspeicher-/Ausrichteinheit verbunden ist, um Steuersignale zu empfangen; und

65 eine Datenoperationsschaltung mit einem Eingang, einem Ausgang und einem Steuereingang, um Datenberechnungen durchzuführen, wobei der Eingang der Datenoperationsschaltung mit dem Ausgang der Abspeicher-/Ausrichteinheit verbunden ist, der Ausgang der Datenoperationseinheit mit dem Rückführ- bzw. Feedbackeingang des Schalters verbunden ist und der Steuereingang der Datenoperationslogik verbunden ist, um Steuersignale zu empfangen.

70 11. Dynamisch rekonfigurierbare Verarbeitungseinheit nach Anspruch 10, bei welcher die Abspeicher-/Ausrichteinheit rekonfigurierbar ist und selektiv als eine aus der Gruppe von einem Speicher mit wahlfreiem Zugriff bzw. ein RAM und einem gepipelineten Register bzw. Fließbandregister konfiguriert werden kann, und zwar in Antwort auf Steuersignale für eine entsprechende Instruktionssatz-Architektur.

DE 196 14 991 A1

12. Dynamisch rekonfigurierbare Verarbeitungseinheit nach Anspruch 10, bei welcher entsprechend Signa-
len die Datenoperationseinheit rekonfigurierbar ist und selektiv als eine aus der Gruppe von einer arithme-
tische Logikeinheit und einer gepipelneten Funktionseinheit bzw. Fließband-Funktionseinheit konfiguriert
werden kann, und zwar in Antwort auf Steuersignale für eine entsprechende Instruktionssatz-Architektur. 5

13. Dynamisch rekonfigurierbare Verarbeitungseinheit nach Anspruch 1, bei welcher die veränderbare
interne Hardware-Organisation der rekonfigurierbaren Verarbeitungseinheit eine Adressen-Operations-
einheit mit einem Steuereingang, einem Adresseneingang und einem Ausgang umfaßt, um Operationen
bezüglich Adressen durchzuführen, wobei der Adresseneingang mit dem Datenport eines Speichers ver-
bunden ist und der Ausgang der Adressen-Operationseinheit mit einem Adresseneingang des Speichers
verbunden ist, und der Steuereingang der Adressen-Operationseinheit verbunden ist, um Steuersignale zu
empfangen. 10

14. Dynamisch rekonfigurierbare Verarbeitungseinheit nach Anspruch 13, bei welcher die Adressen-Opera-
tionseinheit folgendes umfaßt:
einen Schalter mit einem Datenport, einen Steuereingang, einen Rückführ- bzw. Feedbackeingang und
einen Ausgang, um selektiv Adressen zwischen dem Datenport, dem Rückführ- bzw. Feedbackeingang und
dem Ausgang zu leiten bzw. zu führen, und zwar in Antwort auf ein Steuersignal, das bei dem Steuereingang
empfangen wurde, wobei der Datenport des Schalters mit dem Datenport des Speichers verbunden ist;
eine Abspeicher-/Zähleinheit mit einem Eingang, einem Ausgang und einem Steuereingang, um Daten zu
speichern, wobei der Eingang der Abspeicher-/Zähleinheit mit dem Ausgang des Schalters verbunden ist,
der Steuereingang der Abspeicher-/Zähleinheit verbunden ist, um Steuersignale zu empfangen; und
eine Adressen-Operationsschaltung mit einem Eingang, einem Ausgang und einem Steuereingang, um
Adressberechnungen durchzuführen, wobei der Eingang der Adressen-Operationsschaltung mit dem
Ausgang der Abspeicher-/Zähleinheit verbunden ist, der Ausgang der Adressen-Operationsschaltung mit dem
Rückführ- bzw. Feedbackeingang des Schalters verbunden ist, und der Steuereingang der Adressen-
Operationseinheit verbunden ist, um Steuersignale zu empfangen. 15

15. Dynamisch rekonfigurierbare Verarbeitungseinheit nach Anspruch 14, bei welcher die Abspeicher-/
Zähleinheit rekonfigurierbar ist und selektiv als etwas aus der Gruppe konfiguriert werden kann, die ein
Speicher mit wahlfreiem Zugriff bzw. ein RAM und ein Register aufweist, und zwar in Antwort auf Signale,
die an dem Steuereingang der Abspeicher-/Zähleinheit empfangen werden. 20

16. Dynamisch rekonfigurierbare Verarbeitungseinheit nach Anspruch 14, bei welcher die Adressen-Opera-
tionsschaltung als eine aus der Gruppe von einem Register und einem Register und einer arithmetischen
Einheit rekonfigurierbar ist, und zwar in Antwort auf Signale, die bei dem Steuereingang der Adressen-
Operationsschaltung empfangen werden. 25

17. System zur dynamisch rekonfigurierbaren Steuerung, das folgendes umfaßt:
eine erste rekonfigurierbare Verarbeitungseinheit zur Ausführung von Programminstruktionen, um Daten
zu verarbeiten, wobei die erste rekonfigurierbare Verarbeitungseinheit einen Eingang, einen Ausgang und
eine veränderbare interne Hardware-Organisation aufweist, die selektiv während der Ausführung einer
Sequenz von Programminstruktionen veränderbar ist; 30

eine erste Kommunikationsvorrichtung mit einem Eingang, einem Ausgang, einem ersten Datenport und
einem zweiten Datenport, um Daten zu und von der ersten rekonfigurierbaren Verarbeitungseinheit zu
übertragen, wobei der Eingang der ersten Kommunikationsvorrichtung mit dem Ausgang der ersten
rekonfigurierbaren Verarbeitungseinheit verbunden ist, und der Ausgang der ersten Kommunikationsvor-
richtung mit dem Eingang der ersten rekonfigurierbaren Verarbeitungseinheit verbunden ist. 35

18. System nach Anspruch 17, das weiter folgendes umfaßt:
eine zweite rekonfigurierbare Verarbeitungseinheit zur Ausführung von Programminstruktionen, um Da-
ten zu verarbeiten, wobei die zweite rekonfigurierbare Verarbeitungseinheit einen Eingang, einen Ausgang
und eine veränderbare interne Hardware-Organisation umfaßt, die selektiv während der Ausführung einer
Sequenz von Programminstruktionen veränderbar ist; 40

eine zweite Kommunikationsvorrichtung mit einem Eingang, einem Ausgang, einem ersten Datenport und
einem zweiten Datenport, um Daten zu und von der zweiten rekonfigurierbaren Verarbeitungseinheit zu
übertragen, wobei der Eingang der zweiten Kommunikationsvorrichtung mit dem Ausgang der zweiten
rekonfigurierbaren Verarbeitungseinheit verbunden ist, und der Ausgang der zweiten Kommunikationsvor-
richtung mit dem Eingang der zweiten rekonfigurierbaren Verarbeitungseinheit verbunden ist; und
eine Verbindungseinrichtung zum Leiten von Daten und mit einer Vielzahl von Kommunikationskanälen,
wobei sowohl der erste Datenport der ersten Kommunikationsvorrichtung als auch der zweite Datenport
der ersten Kommunikationsvorrichtung und der erste Datenport der zweiten Kommunikationsvorrichtung
und der zweite Datenport der zweiten Kommunikationsvorrichtung mit einem der Vielzahl von Kommuni-
kationskanälen verbunden ist. 45

19. System nach Anspruch 18, bei welchem die erste rekonfigurierbare Verarbeitungseinheit dynamisch
unabhängig von der Rekonfiguration der zweiten rekonfigurierbaren Verarbeitungseinheit rekonfigurier-
bar ist. 50

20. System Anspruch 18, das weiter folgendes aufweist:
eine dritte rekonfigurierbare Verarbeitungseinheit zur Ausführung von Programminstruktionen, um Daten
zu verarbeiten, wobei die dritte rekonfigurierbare Verarbeitungseinheit einen Eingang, einen Ausgang und
eine veränderbare interne Hardware-Organisation aufweist, die selektiv während der Ausführung einer
Sequenz von Programminstruktionen veränderbar ist; und 55

eine dritte Kommunikationsvorrichtung mit einem Eingang, einem Ausgang, einem ersten Datenport und
einem zweiten Datenport, um Daten zu und von der dritten rekonfigurierbaren Verarbeitungseinheit zu

DE 196 14 991 A1

übertragen, wobei der Eingang der dritten Kommunikationsvorrichtung mit dem Ausgang der dritten rekonfigurierbaren Verarbeitungseinheit verbunden ist, der Ausgang der dritten Kommunikationsvorrichtung mit dem Eingang der dritten rekonfigurierbaren Verarbeitungseinheit verbunden ist, sowohl der erste Datenport der dritten Kommunikationsvorrichtung als auch der zweite Datenport der dritten Kommunikationsvorrichtung mit einer der Vielzahl von Kommunikationskanälen der Verbindungseinrichtung verbunden ist.

5 21. System nach Anspruch 17, das weiter folgendes aufweist:
eine nicht-rekonfigurierbare Verarbeitungseinheit mit einer vorbestimmten Architektur, um ein Programm
10 von Instruktionen auszuführen, das aus einem einzigen Instruktionssatz ausgebildet ist, wobei die nicht-re-
konfigurierbare Verarbeitungseinheit einen Eingang und einen Ausgang umfaßt; und
eine zweite Kommunikationsvorrichtung mit einem Eingang, einem Ausgang, einem ersten Datenport und
15 einem zweiten Datenport, um Daten zu und von der nicht-rekonfigurierbaren Verarbeitungseinheit zu
übertragen, wobei der Eingang der dritten Kommunikationsvorrichtung mit dem Ausgang der nicht-rekon-
figurierbaren Verarbeitungseinheit verbunden ist, der Ausgang der dritten Kommunikationsvorrichtung
20 mit dem Eingang der nicht-rekonfigurierbaren Verarbeitungseinheit verbunden ist; und
eine Verbindungseinrichtung zum Leiten von Daten mit einer Vielzahl von Kommunikationskanälen, wobei
sowohl der erste Datenport der ersten Kommunikationsvorrichtung als auch der zweite Datenport der
ersten Kommunikationsvorrichtung und der erste Datenport der zweiten Kommunikationsvorrichtung und
25 der zweite Datenport der zweiten Kommunikationsvorrichtung mit einem der Vielzahl von Kommunika-
tionskanälen verbunden ist.

22. System nach Anspruch 17, das weiter folgendes aufweist:
eine I/O-Vorrichtung mit einem Eingang und einem Ausgang; und
eine I/O-Kommunikationsvorrichtung mit einem Eingang, einem Ausgang, einem ersten Datenport und
30 einem zweiten Datenport, um Daten zu und von der I/O-Vorrichtung zu übertragen, wobei der Eingang der
I/O-Kommunikationsvorrichtung mit dem Ausgang der I/O-Vorrichtung verbunden ist, der Ausgang der
I/O-Kommunikationsvorrichtung mit dem Eingang der I/O-Vorrichtung verbunden ist; und
eine Verbindungseinrichtung zum Leiten von Daten mit einer Vielzahl von Kommunikationskanälen, wobei
sowohl der erste Datenport der ersten Kommunikationsvorrichtung als auch der zweite Datenport der
ersten Kommunikationsvorrichtung und der erste Datenport der I/O-Kommunikationsvorrichtung und der
35 zweite Datenport der I/O-Kommunikationsvorrichtung mit einem der Vielzahl von Kommunikationskanä-
len verbunden ist.

23. System nach Anspruch 17, das weiter eine Master-Zeitbasiseinheit mit einem Ausgang umfaßt, um ein
Master-Zeitsteuersignal zu der ersten rekonfigurierbaren Verarbeitungseinheit zu liefern.

24. System nach Anspruch 23, bei welchem die erste rekonfigurierbare Verarbeitungseinheit weiter eine
lokale Zeitbasis-Einheit mit einem Eingang und einem Ausgang umfaßt, um ein lokales Zeitsteuersignal von
35 dem Master-Zeitsteuersignal zu erzeugen, wobei der Eingang der lokalen Zeitbasis-Einheit mit dem Aus-
gang der Master-Zeitbasiseinheit verbunden ist und der Ausgang der lokalen Zeitbasiseinheit mit einem
Zeitsteuereingang der rekonfigurierbaren Verarbeitungseinheit verbunden ist.

25. System nach Anspruch 17, bei welchem die erste rekonfigurierbare Verarbeitungseinheit aus einer
40 programmierbaren Logikvorrichtung aufgebaut ist.

26. System nach Anspruch 25, bei welchem die programmierbare Logikvorrichtung ein erstes feldprogram-
mierbares Gatter-Array ist, das eine Vielzahl von konfigurierbaren Logikblöcken, eine Vielzahl von pro-
grammierbaren I/O-Blöcken und eine Vielzahl von programmierbaren Verbindungsstrukturen und Daten-
speicher-Systemelementen enthält.

45 27. System nach Anspruch 17, bei welchem das System weiter einen Speicher umfaßt, der einen ersten
Konfigurationsdatensatz speichert, der einer ersten Instruktionssatz-Architektur für einen seriellen Instruktionsprozessor entspricht, und einem zweiten Konfigurationsdatensatz, der einer zweiten Instruktionssatz-
Architektur für einen parallelen Instruktionsprozessor entspricht, umfaßt und bei welchem die erste rekon-
figurierbare Verarbeitungseinheit selektiv als eine aus der Gruppe von einem seriellen Instruktionsprozes-
50 sor und einem parallelen Instruktionsprozessor konfiguriert wird, und zwar in Antwort auf Signale von dem
Speicher, wobei die erste rekonfigurierbare Verarbeitungseinheit mit dem Speicher verbunden ist.

55 28. System nach Anspruch 27, bei welchem die erste rekonfigurierbare Verarbeitungseinheit mit dem
Speicher über eine Vielzahl von Signalleitungen verbunden ist und die erste Anzahl von der Vielzahl von
Signalleitungen Adressenleitungen ausbildet, eine zweite Anzahl von der Vielzahl von Signalleitungen
Steuerleitungen ausbildet und eine dritte Anzahl der Vielzahl von Signalleitungen Datenleitungen ausbildet,
wobei die erste Anzahl, die zweite Anzahl und dritte Anzahl der Vielzahl von Signalleitungen rekonfigurier-
bar sind und gemäß einem Konfigurationsdatensatz gesetzt bzw. eingestellt werden, der durch die erste
rekonfigurierbare Verarbeitungseinheit verwendet wird.

60 29. System nach Anspruch 17, bei welchem die veränderbare interne Hardware-Organisation der ersten
rekonfigurierbaren Verarbeitungseinheit eine rekonfigurierbare Instruktionsabrufeinheit mit einem ersten
Dateneingang, einem ersten Steuerausgang und einem zweiten Steuerausgang umfaßt, um Instruktions-
Ausführungsoperationen innerhalb der ersten rekonfigurierbaren Verarbeitungseinheit sequentiell zu ord-
nen, wobei der Dateneingang mit einem Datenport eines Speichers verbunden ist.

65 30. System nach Anspruch 28, bei welchem die Instruktionsabrufeinheit einen Architekturbeschreibungsspeicher
mit einem Ausgang umfaßt, wobei der Architekturbeschreibungsspeicher einen Satz von Architek-
turbeschreibungssignalen einschließlich eines Interruptantwortsignals speichert, das eine Art und Weise
spezifiziert, in der die erste rekonfigurierbare Verarbeitungseinheit auf ein Interruptsignal antwortet, wenn
sie konfiguriert ist, um eine Instruktionssatz-Architektur zu realisieren bzw. zu implementieren.

31. System nach Anspruch 30, bei welchem die Instruktionsabrufeinheit weiter folgendes umfaßt:
 einen Instruktionszustandssequenzer bzw. eine Instruktionszustandsfolgesteuereinheit mit einem Eingang und einem Ausgang, um einen Instruktionsausführungszyklus mit einem Instruktionsabruftzustand, einen Instruktionsdecodierzustand,
 einen Instruktionsausführungszustand und einen Schreib-zurück-Zustand zu steuern, wobei der Instruktionsausführungszyklus zu der Ausführung einer Instruktion innerhalb der Instruktionssatz-Architektur führt; und 5
 eine Interruptzustandsmaschine mit einem Eingang und einem Ausgang, um ein Übergangssteuersignal zu erzeugen, das einen Zustand innerhalb des Instruktionsausführungszyklus spezifiziert, für den ein Übergang zu einem Interruptservicezustand erlaubt ist, wobei der Eingang der Interruptzustandsmaschine mit dem Ausgang des Architekturbeschreibungsspeichers verbunden ist, der Ausgang der Interruptzustandsmaschine mit dem Eingang des Instruktionszustandssequenzers bzw. der Instruktionszustandsfolgesteuereinheit verbunden ist. 10

32. System nach Anspruch 31, bei welchem die Instruktionsabrufeinheit weiter folgendes aufweist:
 eine programmierbare Abrufsteuereinheit mit einem Eingang und einem Ausgang, um die Operation eines Instruktionspuffers variabler Größe zu steuern, wobei der Eingang der programmierbaren Abrufsteuereinheit mit einem Ausgang der Instruktionszustandsfolgesteuereinheit bzw. des Instruktionszustandssequenzers verbunden ist, um Signale zu empfangen, die die Operation der programmierbaren Abrufsteuereinheit festlegen, wobei der Eingang der programmierbaren Abrufsteuereinheit mit dem Instruktionspuffer verbunden ist; und 15
 eine programmierbare Decodiersteuereinheit mit einem Eingang und einem Ausgang, um die Operation eines Instruktionsdecoders variabler Größe zu steuern, wobei der Eingang der programmierbaren Abrufsteuereinheit mit dem Ausgang des Instruktionszustandssequenzers bzw. der Instruktionszustandsfolgesteuereinheit verbunden ist, um Signale zu empfangen, die die Operation der programmierbaren Decodiersteuereinheit festlegen, wobei der Ausgang der programmierbaren Decodiersteuereinheit mit dem Instruktiondecoder verbunden ist und der Instruktionsdecoder verbunden ist, um Programmstrukturen von dem Instruktionspuffer zu empfangen. 20
 eine programmierbare Decodiersteuereinheit mit einem Eingang und einem Ausgang, um die Operation eines Instruktionsdecoders variabler Größe zu steuern, wobei der Eingang der programmierbaren Abrufsteuereinheit mit dem Ausgang des Instruktionszustandssequenzers bzw. der Instruktionszustandsfolgesteuereinheit verbunden ist, um Signale zu empfangen, die die Operation der programmierbaren Decodiersteuereinheit festlegen, wobei der Ausgang der programmierbaren Decodiersteuereinheit mit dem Instruktiondecoder verbunden ist und der Instruktionsdecoder verbunden ist, um Programmstrukturen von dem Instruktionspuffer zu empfangen. 25

33. System nach Anspruch 17, bei welchem die veränderbare interne Hardware-Organisation der ersten rekonfigurierbaren Verarbeitungseinheit eine rekonfigurierbare Datenoperationseinheit mit einem Datenport und einem Steuereingang umfaßt, um Operationen bezüglich Daten durchzuführen, wobei der Datenport der Datenoperationseinheit mit einem Datenport eines Speichers verbunden ist und der Steuereingang verbunden ist, um Steuersignale zu empfangen. 30

34. System nach Anspruch 33, bei welchem die rekonfigurierbare Datenoperationseinheit folgendes aufweist:
 einen Schalter mit einem Datenport, einem Steuereingang, einem Rückführ- bzw. Feedbackeingang und einem Ausgang, um selektiv Daten zwischen dem Datenport, dem Rückführ- bzw. Feedbackeingang und dem Ausgang zu leiten, wobei der Datenport des Schalters mit dem Datenport des Speichers verbunden ist und der Steuereingang des Schalters mit dem ersten Steuerausgang der Instruktionsabrufeinheit verbunden ist; 35
 eine Abspeicher-/Ausrichteinheit mit einem Eingang, einem Ausgang und einer Steuereinheit, um Daten und Datenberechnungsergebnisse zu speichern, wobei der Eingang der Abspeicher-/Ausrichteinheit mit dem Ausgang des Schalters verbunden ist, der Steuereingang der Speicher-/Ausrichteinheit mit dem ersten Steuerausgang der Instruktionsabrufeinheit verbunden ist; und
 eine Datenoperationsschaltung mit einem Eingang, einem Ausgang und einem Steuereingang, um Datenberechnungen durchzuführen, wobei der Eingang der Datenoperationsschaltung mit dem Ausgang der Abspeicher-/Ausrichteinheit verbunden ist, der Ausgang der Datenoperationsschaltung mit dem Rückführ- bzw. Feedbackeingang des Schalters verbunden ist, und der Steuereingang der Datenoperationsschaltung mit dem ersten Steuerausgang der Instruktionsabrufeinheit verbunden ist. 40
 eine Datenoperationsschaltung mit einem Eingang, einem Ausgang und einem Steuereingang, um Datenberechnungen durchzuführen, wobei der Eingang der Datenoperationsschaltung mit dem Ausgang der Abspeicher-/Ausrichteinheit verbunden ist, der Ausgang der Datenoperationsschaltung mit dem Rückführ- bzw. Feedbackeingang des Schalters verbunden ist, und der Steuereingang der Datenoperationsschaltung mit dem ersten Steuerausgang der Instruktionsabrufeinheit verbunden ist. 45

35. System nach Anspruch 34, bei welchem die Speicher-/Ausrichteinheit als etwas von der Gruppe rekonfigurierbar ist, die einen Speicher mit wahlfreiem Zugriff bzw. ein RAM aufweist und ein gepipelinetes Register bzw. ein Fließbandregister aufweist, und zwar in Antwort auf Steuersignale von dem Speicher, die ein Konfigurationsdatensatz darstellen, der einer ersten Instruktionssatz-Architektur bzw. einer zweiten Instruktionssatz-Architektur entspricht. 50

36. System nach Anspruch 35, bei welchem die Datenoperationseinheit als eine aus der Gruppe von einer arithmetischen Logikeinheit und einer gepipelinierten Funktionseinheit rekonfigurierbar ist, und zwar in Antwort auf Konfigurationssignale aus dem Speicher. 55

37. System nach Anspruch 17, bei welchem die veränderbare interne Hardware-Organisation der ersten rekonfigurierbaren Verarbeitungseinheit eine rekonfigurierbare Adressenoperationseinheit mit einem ersten Steuereingang, einem Adresseneingang und einem Ausgang umfaßt, um Operationen bezüglich Adressen durchzuführen, wobei der Adresseneingang mit dem Datenport eines Speichers verbunden ist und der Ausgang der Adressenoperationseinheit mit einem Adresseneingang des Speichers verbunden ist und der Steuereingang der Adressenoperationseinheit verbunden ist, um Steuersignale zu empfangen. 60

38. System nach Anspruch 37, bei welchem die rekonfigurierbare Adressenoperationseinheit folgendes umfaßt:
 einen Schalter mit einem Datenport, einem Steuereingang, einem Rückführ- bzw. Feedbackeingang und einem Ausgang, um selektiv Adressen zwischen dem Datenport, dem Rückführ- bzw. Feedbackeingang und dem Ausgang zu leiten, wobei der Datenport des Schalters mit dem Datenport des Speichers verbunden ist und der Steuereingang des Schalters mit dem ersten Steuerausgang der Instruktionsabrufeinheit verbunden 65

ist;
5 eine Abspeicher-/Zähleinheit mit einem Eingang, einem Ausgang und einem Steuereingang, um Daten zu speichern, wobei der Eingang der Abspeicher-/Zähleinheit mit dem Ausgang des Schalters verbunden ist, der Steuereingang der Abspeicher-/Zählogik mit dem zweiten Steuerausgang der Instruktionsabrufeinheit verbunden ist; und
10 eine Adressenoperationsschaltung mit einem Eingang, einem Ausgang und einem Steuereingang, um Adressenberechnungen durchzuführen, wobei der Eingang der Adressenoperationsschaltung mit dem Ausgang der Abspeicher-/Zähleinheit verbunden ist, der Ausgang der Adressenoperationsschaltung mit dem Rückführ- bzw. Feedbackeingang des Schalters verbunden ist, und der Steuereingang der Adressenoperations-
15 einheit mit dem zweiten Steuerausgang der Instruktionsabrufeinheit verbunden ist.
39. System nach Anspruch 38, bei welchem die Abspeicher-/Zähleinheit rekonfigurierbar ist und selektiv als eine aus der Gruppe von einem Speicher mit wahlfreiem Zugriff bzw. einem RAM und einem Register konfiguriert werden kann, und zwar in Antwort auf Signale, die am Steuereingang der Abspeicher-/Zähleinheit empfangen werden.
40. System nach Anspruch 38, bei welchem die Adressenoperationsschaltung als eine aus der Gruppe von einem Register und einem Register und einer arithmetischen Einheit rekonfigurierbar ist, und zwar in Antwort auf Signale, die an dem Steuereingang der Adressenoperationsschaltung empfangen werden.
41. System nach Anspruch 17, bei welchem die erste rekonfigurierbare Verarbeitungseinheit folgendes umfaßt:
20 eine rekonfigurierbare Instruktionsabrufeinheit mit einem Dateneingang, einem ersten Steuereingang und einem zweiten Steuerausgang, um Instruktionsausführungsoperationen innerhalb der ersten rekonfigurierbaren Verarbeitungseinheit sequentiell zu ordnen, wobei der Dateneingang mit dem Datenport eines Speichers verbunden ist;
25 eine rekonfigurierbare Datenoperationseinheit mit einem Datenport und einem Steuereingang, um Operationen bezüglich Daten durchzuführen, wobei der Datenport der Datenoperationseinheit mit dem Datenport des Speichers verbunden ist und der Steuereingang mit dem ersten Steuerausgang der Instruktionsabrufeinheit verbunden ist; und
30 eine rekonfigurierbare Adressenoperationseinheit mit einem Steuereingang, einem Adresseneingang und einem Ausgang, um Operationen bezüglich Adressen durchzuführen, wobei der Steuereingang der Adressenoperationseinheit mit dem zweiten Steuerausgang der Instruktionsabrufeinheit verbunden ist, der Adresseneingang mit dem Datenport des Speichers verbunden ist und der Ausgang der Adressenoperationseinheit mit einem Adresseneingang des Speichers verbunden ist.
35 42. System nach Anspruch 41, bei welchem die rekonfigurierbare Instruktionsabrufeinheit, die rekonfigurierbare Datenoperationseinheit und die rekonfigurierbare Adressenoperationseinheit während der Ausführung eines Befehls durch die erste rekonfigurierbare Verarbeitungseinheit rekonfigurierbar sein kann.
43. System zum Koppeln bzw. Verbinden eines ersten Prozessors bzw. einer ersten Verarbeitungseinrichtung mit einem zweiten Prozessor bzw. einer zweiten Verarbeitungseinrichtung, wobei das System folgendes umfaßt:
40 eine erste Kommunikationsvorrichtung mit einem Eingang, einem Ausgang, einem ersten Datenport und einem zweiten Datenport, um Daten zu und von dem ersten Prozessor zu übertragen, wobei der Eingang und der Ausgang der ersten Kommunikationsvorrichtung mit dem ersten Prozessor verbunden ist; und
45 eine zweite Kommunikationsvorrichtung mit einem Eingang, einem Ausgang, einem ersten Datenport und einem zweiten Datenport, um Daten zu und von dem zweiten Prozessor zu übertragen, wobei der Eingang und der Ausgang der zweiten Kommunikationsvorrichtung mit dem zweiten Prozessor verbunden ist; und
50 eine Verbindungseinrichtung zum Bereitstellen einer Punkt-zu-Punkt-Parall-Datenwegeleitung mit einem ersten und einem zweiten Kommunikationskanal, wobei der erste Datenport der ersten Kommunikationsvorrichtung und der erste Datenport der zweiten Kommunikationsvorrichtung mit dem ersten Kommunikationskanal verbunden ist und der zweite Datenport der ersten Kommunikationsvorrichtung und der zweite Datenport der zweiten Kommunikationsvorrichtung mit dem zweiten Kommunikationskanal verbunden ist.
55 44. System nach Anspruch 43, bei welchem die erste und die zweite Kommunikationsvorrichtung jeweils als ein feldprogrammierbares Gatter-Array aufgebaut bzw. konstruiert sind.
45. System nach Anspruch 43, bei welchem die erste Kommunikationsvorrichtung weiter folgendes aufweist:
55 eine Schnittstellen- und Steuereinheit mit einem ersten Datenport, einem zweiten Datenport und einem Steuerport, um Daten und Befehle bzw. Kommandos zu empfangen und zu senden, und zwar zu und von dem ersten Prozessor, wobei der erste Datenport mit dem ersten Prozessor zum Senden und Empfangen von Daten verbunden ist und der Steuerport mit dem ersten Prozessor zum Senden und Empfangen von Befehlen bzw. Kommandos verbunden ist;
60 eine erste Verbindungseinheit mit einem Eingang, einem Ausgang, einem Port und einer einzigen Verbindungsadresse, wobei der Port der ersten Verbindungseinheit mit dem zweiten Datenport der Schnittstellen- und Steuereinheit verbunden ist, und der Eingang und Ausgang der ersten Verbindungseinheit mit Knoten des ersten Kommunikationskanals verbunden sind; und
65 eine zweite Verbindungseinheit mit einem Eingang, einem Ausgang, einem Port und einer einzigen Verbindungsadresse, wobei die zweite Verbindungseinheit mit dem zweiten Datenport der Schnittstellen- und Steuereinheit verbunden ist und der Eingang und der Ausgang der zweiten Verbindungseinheit mit Knoten des zweiten Kommunikationskanals verbunden sind.

46. System nach Anspruch 45, bei welchem die erste Verbindungseinheit folgendes aufweist:
einen Adressendecoder mit einem Eingang und einem ersten und zweiten Ausgang, um selektiv Daten zu
einem der ersten und zweiten Ausgänge zu leiten, und zwar in Antwort auf eine Adresse, die als ein Teil
eines Nachrichtenpakets ausgebildet ist, wobei der Eingang des Adressendecoders mit dem ersten Kommuni-
kationskanal verbunden ist;

5

einen ersten Puffer zum Speichern von Daten, wobei der erste Puffer einen Eingang und einen Ausgang
aufweist, wobei der Eingang des ersten Puffers mit dem ersten Ausgang des Adressendecoders verbunden
ist und der Ausgang mit dem Interface und der Steuereinheit verbunden ist;

10

einen zweiten Puffer zum Speichern von Daten, wobei der zweite Puffer einen Eingang und einen Ausgang
aufweist, wobei der Eingang des zweiten Puffers mit dem zweiten Ausgang des Adressendecoders verbun-
den ist;

einen dritten Puffer zum Speichern von Daten, wobei der dritte Puffer einen Eingang und einen Aus-
gang aufweist, wobei der Eingang des dritten Puffers mit der Schnittstellen- und Steuereinheit verbunden
ist;

einen Multiplexer mit einem ersten Eingang, einem zweiten Eingang, einem Steuereingang und einem
Ausgang, um selektiv Daten von dem ersten Eingang oder dem zweiten Eingang in Antwort auf ein
Steuersignal zu dem Steuereingang weiterzugeben, wobei der erste Eingang mit dem Ausgang des zweiten
Puffers verbunden ist, der zweite Eingang mit dem dritten Puffer verbunden ist, der Steuereingang mit der
Schnittstellen- und Steuerlogik verbunden ist und der Ausgang des Multiplexers mit dem ersten Kommuni-
kationskanal verbunden ist.

15

47. System nach Anspruch 45, bei welchem

20

der erste Prozessor mit der Schnittstellen- und Steuereinheit über einen Speicher verbunden ist, der eine
Vielzahl von Signalleitungen aufweist, und wobei eine erste Anzahl der Vielzahl von Signalleitungen
Adressenleitungen ausbilden, eine zweite Anzahl der Vielzahl von Signalleitungen Steuerleitungen ausbil-
den und eine dritte Anzahl der Vielzahl von Signalleitungen Datenleitungen ausbilden; und
der erste Prozessor und die Schnittstellen- und Steuereinheit derartig rekonfigurierbar sind, daß die erste
Anzahl, die zweite Anzahl und die dritte Anzahl der Vielzahl von Signalleitungen gemäß eines Konfigura-
tionsdatensatzes, der durch den ersten Prozessor und die Schnittstellen- und Steuereinheit genutzt bzw.
verwendet wird, eingestellt bzw. gesetzt werden kann.

25

48. System nach Anspruch 45, bei welchem

30

der erste Prozessor mit der Schnittstellen- und Steuereinheit über einen Speicher, der eine Vielzahl von
Signalleitungen aufweist, verbunden ist und wobei eine erste Anzahl der Vielzahl von Signalleitungen
Adressenleitungen ausbilden, eine zweite Anzahl der Vielzahl von Signalleitungen Steuerleitungen ausbil-
den und eine dritte Anzahl der Vielzahl von Signalleitungen Datenleitungen ausbilden; und
der erste Prozessor und die Schnittstellen- und Steuereinheit derartig rekonfigurierbar sind, daß die erste
Anzahl, die zweite Anzahl und die dritte Anzahl der Vielzahl von Signalleitungen gemäß eines Konfigura-
tionsdatensatzes eingestellt werden können, der durch den ersten Prozessor und die Schnittstellen- und
Steuereinheit verwendet bzw. genutzt wird.

35

49. System nach Anspruch 45, bei welchem

40

der erste Prozessor mit der Schnittstellen- und Steuereinheit über einen Speicher, der eine Vielzahl von
Signalleitungen aufweist, verbunden ist, und wobei eine erste Anzahl der Vielzahl von Signalleitungen
Adressenleitungen ausbilden, eine zweite Anzahl der Vielzahl von Signalleitungen Steuerleitungen ausbil-
den und eine dritte Anzahl der Vielzahl von Signalleitungen Datenleitungen ausbilden; und
die Schnittstellen- und Steuereinheit derartig rekonfigurierbar ist, daß die erste Anzahl, die zweite Anzahl
bzw. die dritte Anzahl der Vielzahl von Signalleitungen eingestellt werden kann, um mit einer Anzahl von
Adressen-, Steuer- bzw. Datenleitungen zusammenzupassen, die durch den ersten Prozessor verwendet
werden, um auf den Speicher zuzugreifen und diesen zu steuern.

45

50. System nach Anspruch 45, bei welchem die Schnittstellen- und Steuereinheit eine Nachricht in einen
Befehl bzw. in ein Kommando und in Daten, die von dem ersten Prozessor verwendet werden können,
umwandelt und Kommandos bzw. Befehle und Daten von dem ersten Prozessor in eine Nachricht zur
Übertragung über die Verbindungseinrichtung bündelt bzw. packetiert.

50

51. Verfahren zur Erzeugung von Instruktionen, die von einem rekonfigurierbaren Computer aus einer
Vielzahl von Aussagen hohen Niveaus ausführbar sind, wobei das Verfahren die folgenden Schritte auf-
weist:

eine Vielzahl von Sätzen von Regeln zum Übersetzen von Aussagen hohen Niveaus in Instruktionen, die
durch einen rekonfigurierbaren Computer ausführbar sind, wird bereitgestellt;

55

einer der Vielzahl von Sätzen von Regeln wird als der gegenwärtige Satz von Regeln ausgewählt, der
verwendet werden soll, um Aussagen hohen Niveaus in Instruktionen zu übersetzen, die durch einen
rekonfigurierbaren Computer ausführbar sind;

eine Aussage hohen Niveaus wird ausgewählt;

60

ob die gewählte Aussage hohen Niveaus eine Rekonfigurationsanweisung ist, wird bestimmt;

der gegenwärtige Satz von Regeln, der zum Übersetzen von Aussagen hohen Niveaus verwendet werden
soll, wird in einen Satz von Regeln geändert, der in der Rekonfigurationsanweisung spezifiziert ist, falls die
gewählte Aussage hohen Niveaus eine Rekonfigurationsanweisung ist; und

65

die gewählte Aussage hohen Niveaus wird in wenigstens eine Instruktion übersetzt, die durch eine rekonfi-
gurierbaren Computer ausführbar ist, indem der gegenwärtige Satz von Regeln verwendet wird.

52. Verfahren nach Anspruch 51, bei welchem jeder der Vielzahl von Sätzen von Regeln zum Übersetzung
von Aussagen hohen Niveaus in Instruktionen, die durch den rekonfigurierbaren Computer ausführbar sind,

einer unterschiedlichen Instruktionssatz-Architektur entspricht.

53. Verfahren nach Anspruch 51, das weiter folgende Schritte aufweist:
die Rekonfigurationsanweisung wird in eine Rekonfigurationsaussage mittleren Niveaus übersetzt, falls die gewählte Aussage hohen Niveaus eine Rekonfigurationsanweisung ist;
5 die gewählte Aussage wird in eine Aussage mittleren Niveaus übersetzt, falls die gewählte Aussage hohen Niveaus nicht eine Rekonfigurationsanweisung ist;
eine Registerzuweisung wird durchgeführt;
bei welchem der Schritt der Veränderung des gegenwärtigen Satzes von Regeln die folgenden Unterschritte aufweist:
10 eine Aussage mittleren Niveaus wird ausgewählt;
es wird bestimmt, ob die gewählte Aussage mittleren Niveaus eine Rekonfigurationsaussage mittleren Niveaus ist; und
ein Satz von Regeln wird ausgewählt, der der Instruktionssatz-Architektur entspricht, die durch die Rekonfigurationsaussage mittleren Niveaus spezifiziert ist, falls die gewählte Aussage mittleren Niveaus eine Rekonfigurationsaussage mittleren Niveaus ist; und
15 bei welchem der Schritt des Übersetzens der gewählten Aussage hohen Niveaus den Schritt der Erzeugung einer Assemblersprachaussage aus der gewählten Aussage mittleren Niveaus beinhaltet, wobei der gewählte Satz von Regeln verwendet wird, der der Instruktionssatz-Architektur entspricht, die durch die Rekonfigurationsaussage mittleren Niveaus spezifiziert ist.

20 54. Dynamisch rekonfigurierbares Computersystem mit einer rekonfigurierbaren Verarbeitungseinheit, wobei ein Verfahren zum dynamisch rekonfigurierbaren Rechnen folgende Schritte aufweist:
die rekonfigurierbare Verarbeitungseinheit zur Operation gemäß eines ersten Konfigurationsdatensatzes, der einer ersten Instruktionssatz-Architektur entspricht, wird konfiguriert, wobei die Konfiguration der rekonfigurierbaren Verarbeitungseinheit zu einer Hardware-Organisation führt, die die erste Instruktionssatz-Architektur realisiert bzw. implementiert;
ein Instruktionssausführungszyklus der rekonfigurierbaren Verarbeitungseinheit wird unterbrochen; und
die rekonfigurierbare Verarbeitungseinheit gemäß einem zweiten Konfigurationsdatensatz, der einer zweiten Instruktionssatz-Architektur entspricht, wird derart rekonfiguriert daß die Hardware-Organisation der rekonfigurierbaren Verarbeitungseinheit die zweite Instruktionssatz-Architektur realisiert.

25 55. Verfahren nach Anspruch 54, das weiter den Schritt aufweist, wonach während der Ausführung einer Sequenz von Programminstruktionen bestimmt wird, ob Hardware innerhalb der dynamisch rekonfigurierbaren Verarbeitungseinheit rekonfiguriert werden soll.

30 56. Verfahren nach Anspruch 55, das weiter den Schritt beinhaltet, wonach ein neuer Instruktionssausführungszyklus nach dem Schritt einer Rekonfigurierung ausgelöst wird.

35 57. Verfahren nach Anspruch 56, bei welchem der Instruktionssausführungszyklus einer ersten Programminstruktion innerhalb der Sequenz von Programminstruktionen entspricht, und der neue Instruktionssausführungszyklus einer zweiten Programminstruktion innerhalb der Sequenz von Programminstruktionen entspricht.

40 58. Verfahren nach Anspruch 54, das weiter die folgenden Schritte aufweist:
ein erster Satz von Übergangssteuersignalen, der der ersten Instruktionssatz-Architektur entspricht, wird erzeugt, wobei der erste Satz von Übergangssteuersignalen einen Satz unterbrechbarer Zustände innerhalb des Instruktionssausführungszyklus spezifiziert; und
ein zweiter Satz von Übergangssteuersignalen wird erzeugt, der der zweiten Instruktionssatz-Architektur entspricht, wobei der zweite Satz von Übergangssteuersignalen einen Satz unterbrechbarer Zustände innerhalb eines zweiten Instruktionssausführungszyklus spezifiziert.

45 59. Dynamisch rekonfigurierbarer Computer, der folgendes aufweist:
eine Einrichtung zum Konfigurieren einer rekonfigurierbaren Verarbeitungseinheit mit einem ersten Konfigurationsdatensatz, der einer ersten Instruktionssatz-Architektur entspricht, wobei die Konfiguration der rekonfigurierbaren Verarbeitungseinheit zu einer Hardware-Organisation führt, die die erste Instruktionssatz-Architektur realisiert bzw. implementiert;
eine Einrichtung, um während der Ausführung einer Sequenz von Programminstruktionen zu bestimmen, ob die rekonfigurierbare Verarbeitungseinheit rekonfiguriert werden soll; und
50 eine Einrichtung zum Rekonfigurieren der rekonfigurierbaren Verarbeitungseinheit gemäß eines zweiten Konfigurationsdatensatzes derart, daß die rekonfigurierbare Verarbeitungseinheit eine neue Hardware-Organisation aufweist, die eine zweite Instruktionssatz-Architektur realisiert bzw. implementiert.

55 60. Computer nach Anspruch 59, der weiter folgendes aufweist:
eine Einrichtung, um einen ersten Instruktionssausführungszyklus, der einer ersten Programminstruktion innerhalb einer Sequenz von Programminstruktionen entspricht für den Fall zu unterbrechen, daß die rekonfigurierbare Verarbeitungseinheit rekonfiguriert werden soll; und
eine Einrichtung, um einen zweiten Instruktionssausführungszyklus auszulösen, der einer zweiten Programminstruktion innerhalb der Sequenz von Programminstruktionen entspricht, und zwar nachdem die rekonfigurierbare Verarbeitungseinheit rekonfiguriert worden ist.

60 61. Computer nach Anspruch 59, der weiter eine Einrichtung zur Erzeugung eines Satzes von Übergangssteuersignalen erzeugt, die einer Instruktionssatz-Architektur entsprechen, wobei der Satz von Übergangssteuersignalen einen Satz von unterbrechbaren Zuständen innerhalb eines Instruktionssausführungszyklus spezifiziert.

65 62. Dynamisch rekonfigurierbarer Computer innerhalb einer reprogrammierbaren Logikvorrichtung bzw. innerhalb eines reprogrammierbaren Logikbauelements, wobei der dynamisch rekonfigurierbare Compu-

ter folgendes aufweist:

einen Architekturbeschreibungsspeicher mit einem Ausgang, um Architekturbeschreibungssignale zu speichern, die eine Architektur des rekonfigurierbaren Computers charakterisieren, wobei die Architekturbeschreibungssignale ein Interruptantwortsignal umfassen, das eine Art und Weise spezifiziert, in der der dynamisch rekonfigurierbare Computer auf einen Interrupt antwortet;

5

einen Instruktionszustandssequenzer bzw. eine Instruktionszustandsfolgesteuereinheit mit einem Eingang, um eine Ausführung einer Instruktion durch den rekonfigurierbaren Computer zu steuern; und

10

eine Interruptzustandsmaschine mit einem ersten Eingang, einem zweiten Eingang und einem Ausgang, um ein Übergangssteuersignal zu erzeugen, das spezifiziert bzw. vorgibt, wenn während des Instruktionsausführungszyklus ein Übergang zu einem Interruptservicezustand erlaubt ist, wobei der Eingang der Interruptzustandsmaschine mit dem Ausgang des Architekturbeschreibungsspeichers verbunden ist und der Ausgang der Interruptzustandsmaschine mit dem Eingang des Instruktionszustandssequenzers bzw. der Instruktionszustandsfolgesteuereinheit verbunden ist.

63. Computer nach Anspruch 62, bei welchem

15

ein erster Abschnitt der rekonfigurierbaren Logikvorrichtung gemäß einem ersten Satz von Konfigurationsdaten konfiguriert ist, und zwar derart, daß eine erste Instruktionssatz-Architektur in dem ersten Abschnitt ausgebildet wird;

20

ein zweiter Abschnitt der rekonfigurierbaren Logikvorrichtung gemäß einem zweiten Satz von Konfigurationsdaten derart konfiguriert wird, und zwar derart, daß eine zweite Instruktionssatz-Architektur in dem zweiten Abschnitt ausgebildet wird;

25

der rekonfigurierbare Computer konfiguriert werden kann, um den ersten Abschnitt oder den zweiten Abschnitt zur Verarbeitung zu verwenden; und

die Interruptzustandsmaschine in den Übergangssteuersignalen eine Anzeige des Abschnittes der reprogrammierbaren Logikvorrichtungen bzw. einen Hinweis auf den Abschnitt, auf den der rekonfigurierbare Computer zum Verwenden bzw. zum Gebrauch eingestellt bzw. gesetzt ist, und die Instruktionen, die durch die ausgewählte Instruktionssatz-Architektur ausgeführt werden können, beinhaltet.

30

64. Verfahren zum Verarbeiten von Daten mit einem dynamisch rekonfigurierbaren Computer, der einen Instruktionsausführungszyklus bereitstellt, der einen Instruktionsabrufzustand, einen Instruktionsdecodierzustand, einen Instruktionsausführungszustand und einen Schreib-zurück-Zustand umfaßt, wobei das Verfahren folgende Schritte aufweist:

35

der Computer wird zum Betrieb gemäß einer ersten Instruktionssatz-Architektur konfiguriert, die unterbrechbare bzw. auf ein Interrupt ansprechende Zustände spezifiziert bzw. vorgibt;

ein Interruptsignal wird empfangen;

35

ein Betriebszustand bzw. Operationszustand für den dynamischen rekonfigurierbaren Computer wird bestimmt;

es wird bestimmt, ob der Betriebszustand bzw. Operationszustand des dynamisch rekonfigurierbaren Computers unterbrechbar ist;

das Interruptsignal wird abgearbeitet, falls der dynamisch rekonfigurierbare Computer bestimmt ist, in einem unterbrechbaren Zustand zu sein.

40

65. Verfahren nach Anspruch 64, das weiter die folgenden Schritte aufweist:

ein nächster unterbrechbarer Zustand wird bestimmt, indem der Betrieb bzw. die Operation des dynamisch rekonfigurierbaren Computers unterbrechbar sein kann, falls der dynamisch rekonfigurierbare Computer sich nicht in einem unterbrechbaren Zustand befindet;

45

es wird bestimmt, wenn der dynamisch rekonfigurierbare Computer in den nächsten unterbrechbaren Zustand übergeht, falls der dynamisch rekonfigurierbare Computer sich nicht in einem unterbrechbaren Zustand befindet;

45

das Interruptsignal in dem nächsten unterbrechbaren Zustand wird abgearbeitet bzw. bedient, falls bestimmt ist, daß der dynamisch rekonfigurierbare Computer nicht in einem unterbrechbaren Zustand ist.

66. Verfahren nach Anspruch 64, bei welchem der Schritt des Bedienens bzw. Abarbeitens des Interruptsignals weiter folgende Schritte aufweist:

50

ein Ursprung bzw. Anfang, eine Priorität und eine Interrupthandleradresse bzw. eine Adresse der Interrupthandhabungseinrichtung wird für den Interrupt bestimmt;

es wird bestimmt, ob das Interruptsignal eine Rekonfiguration anzeigt; und

55

Konfigurationsdaten für eine neue Instruktionssatz-Architektur werden im Interruptsteuerregister gespeichert, falls das Interruptsignal Rekonfiguration anzeigt.

67. Verfahren nach Anspruch 64, bei welchem der Schritt, wonach bestimmt wird, ob der Betriebszustand bzw. Operationszustand des dynamisch rekonfigurierbaren Computers unterbrechbar ist, weiter folgende Schritte aufweist:

55

ein Übergangssteuersignal wird empfangen, das der Instruktionssatz-Architektur entspricht, für die der dynamisch rekonfigurierbare Computer konfiguriert worden ist, wobei die Übergangssteuersignale wenigstens einen Zustand spezifizieren, in dem die Ausführung von Instruktionen unterbrechbar ist; und die Zustände, die durch die Übergangssteuersignale spezifiziert sind, werden mit dem Betriebszustand bzw. Operationszustand des dynamisch rekonfigurierbaren Computers verglichen.

60

68. Verfahren nach Anspruch 64, bei welchem der dynamisch rekonfigurierbare Computer gemäß einer Anzahl von verschiedenen Instruktionssatz-Architekturen konfiguriert werden kann, wobei jeder der Instruktionssatz-Architekturen ein Satz von Übergangssteuersignalen liefert, die wenigstens einen unterbrechbaren Betriebszustand einer Instruktionsausführung festlegen.

65

69. Verfahren nach Anspruch 68, bei welchem jede Instruktionssatz-Architektur einen rekonfigurierbaren

DE 196 14 991 A1

Interruptmechanismus umfaßt, der durch Modifikation des Satzes von Übergangssteuersignalen bereitgestellt wird, die der Instruktionssatz-Architektur zugeordnet sind.

Hierzu 23 Seite(n) Zeichnungen

5

10

15

20

25

30

35

40

45

50

55

60

65

- Leerseite -

This page Blank (uspto)

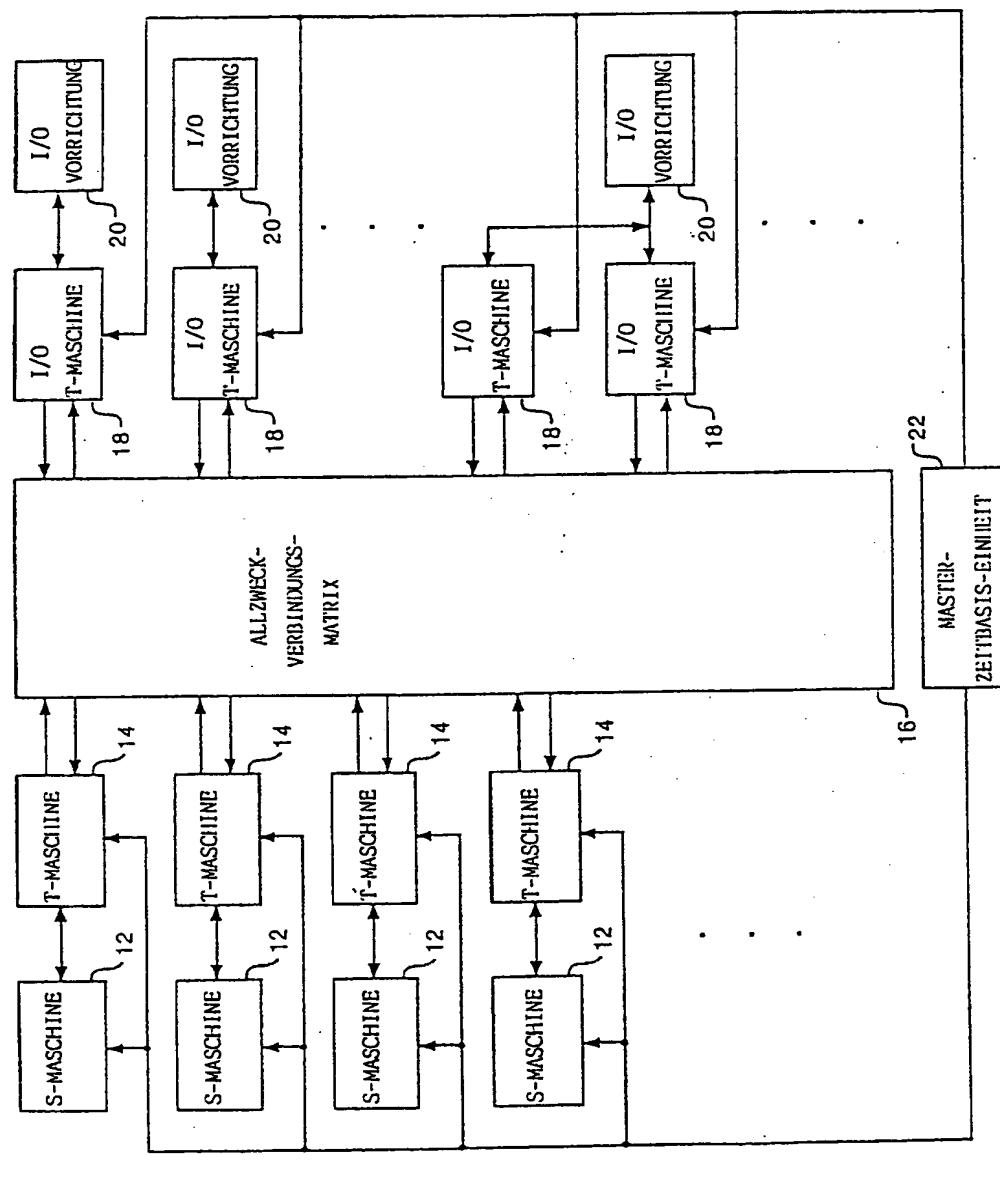


FIG. 1

10

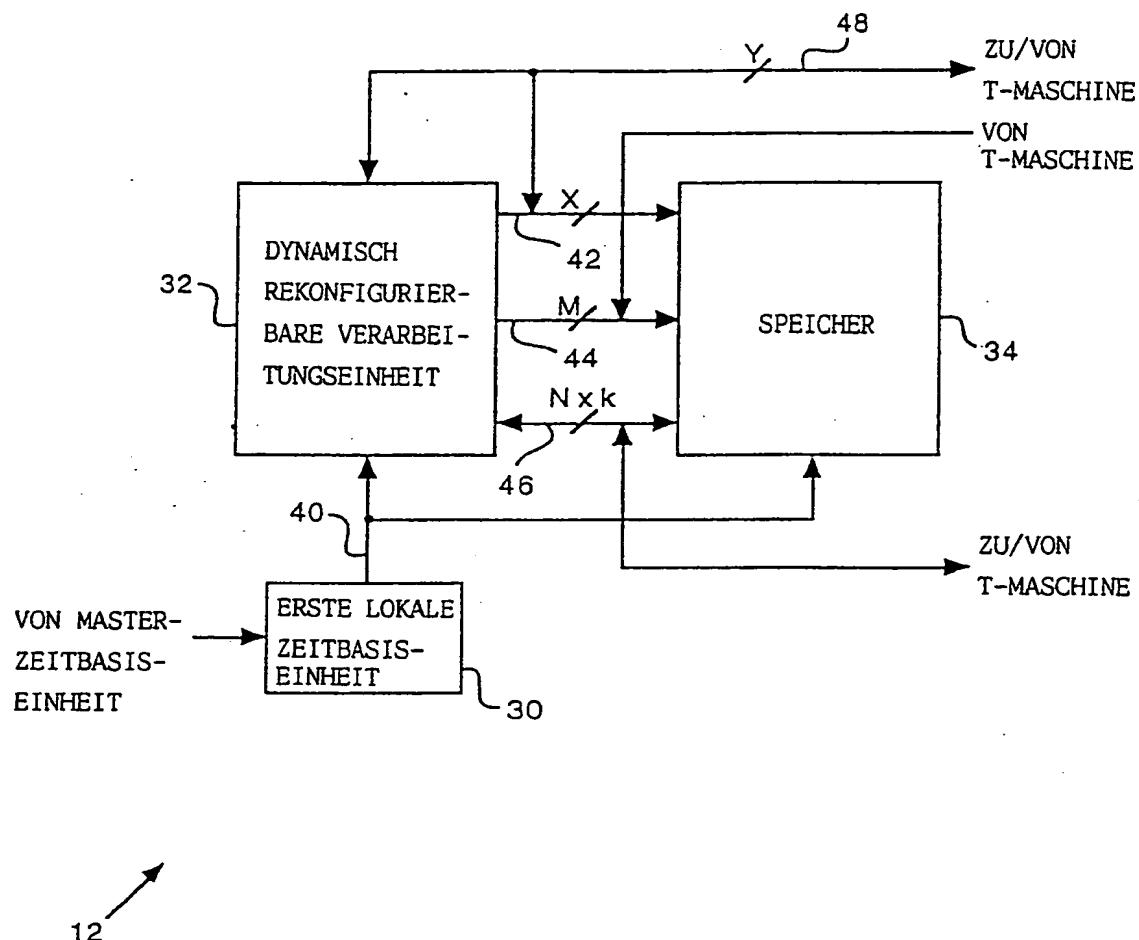


FIG. 2

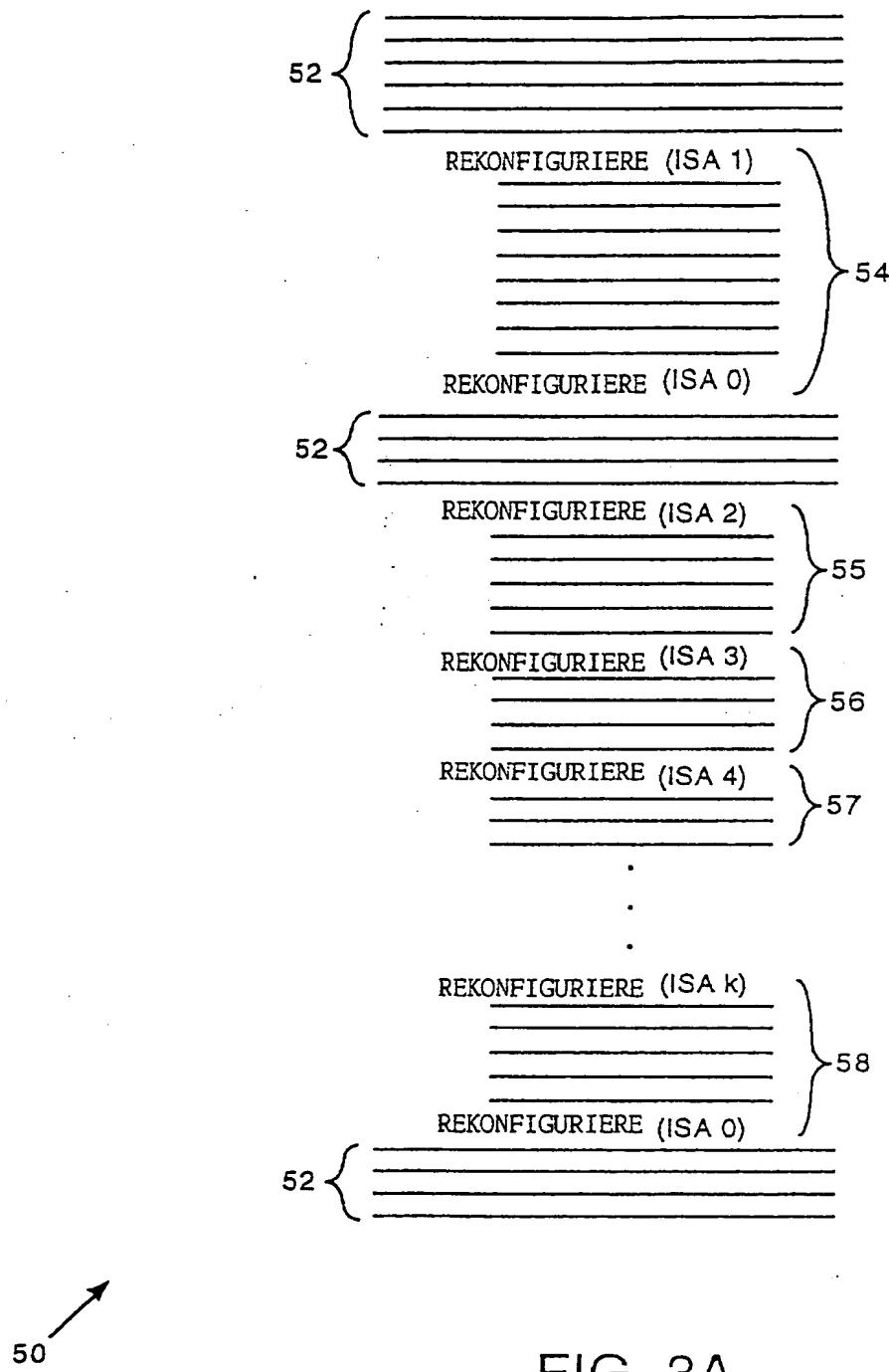


FIG. 3A

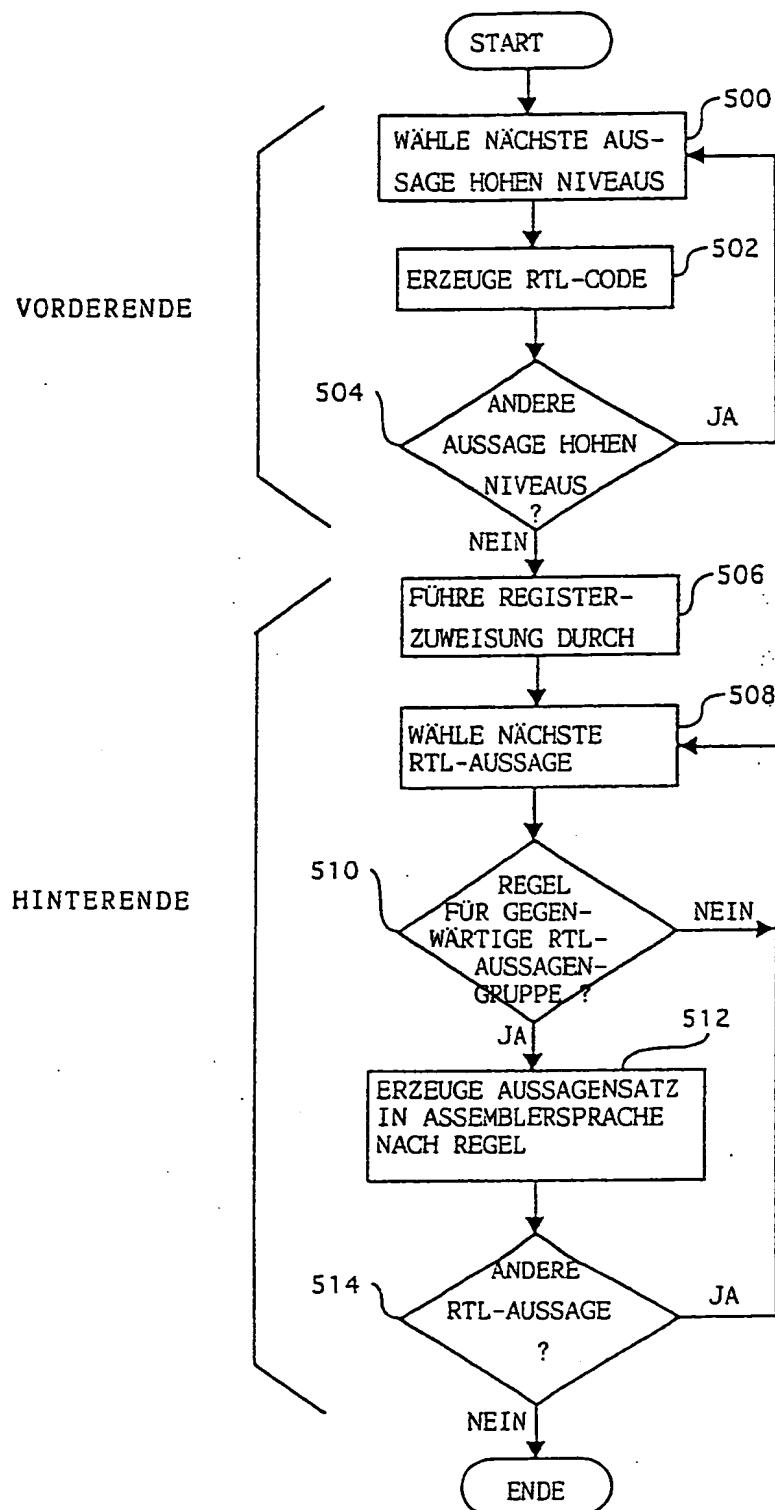


FIG. 3B

STAND DER TECHNIK

602 043/434

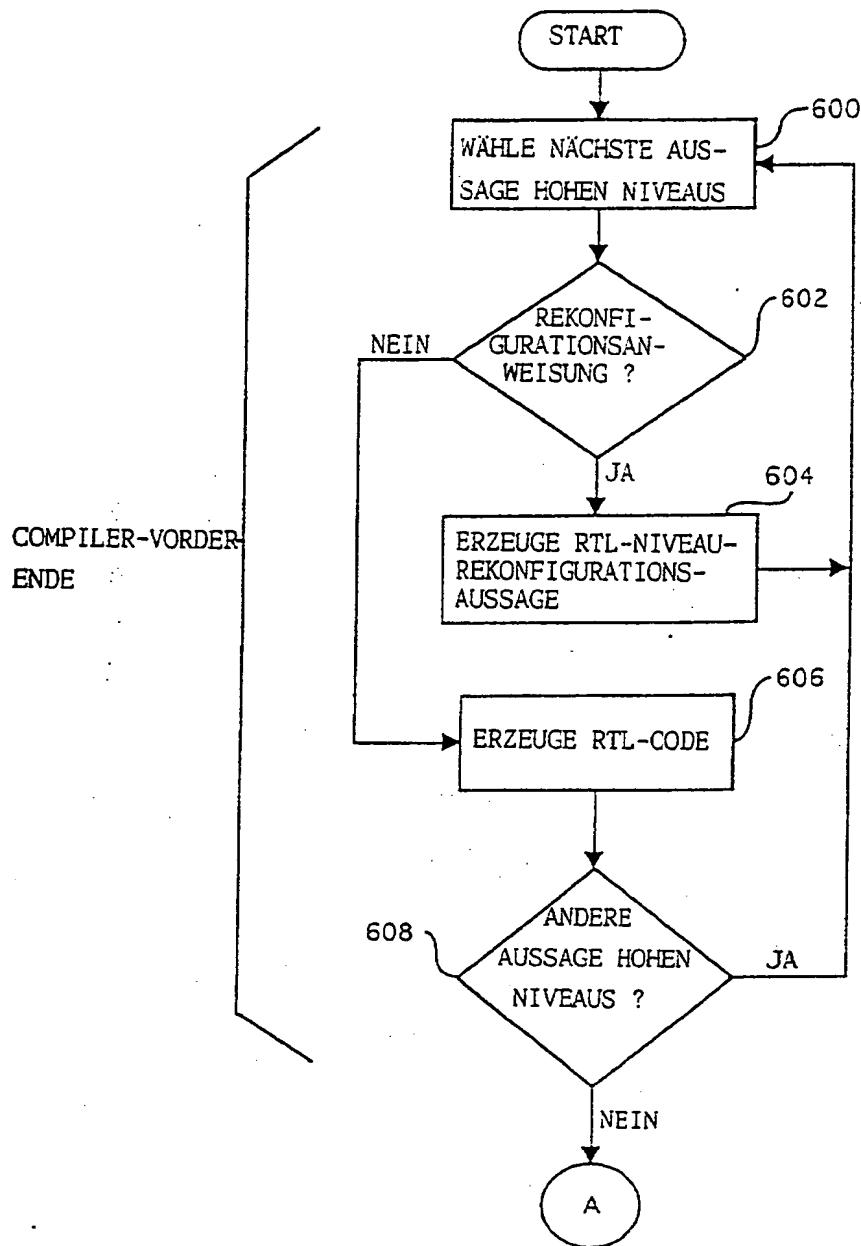


FIG. 3C

COMPILER-HINTERENDE

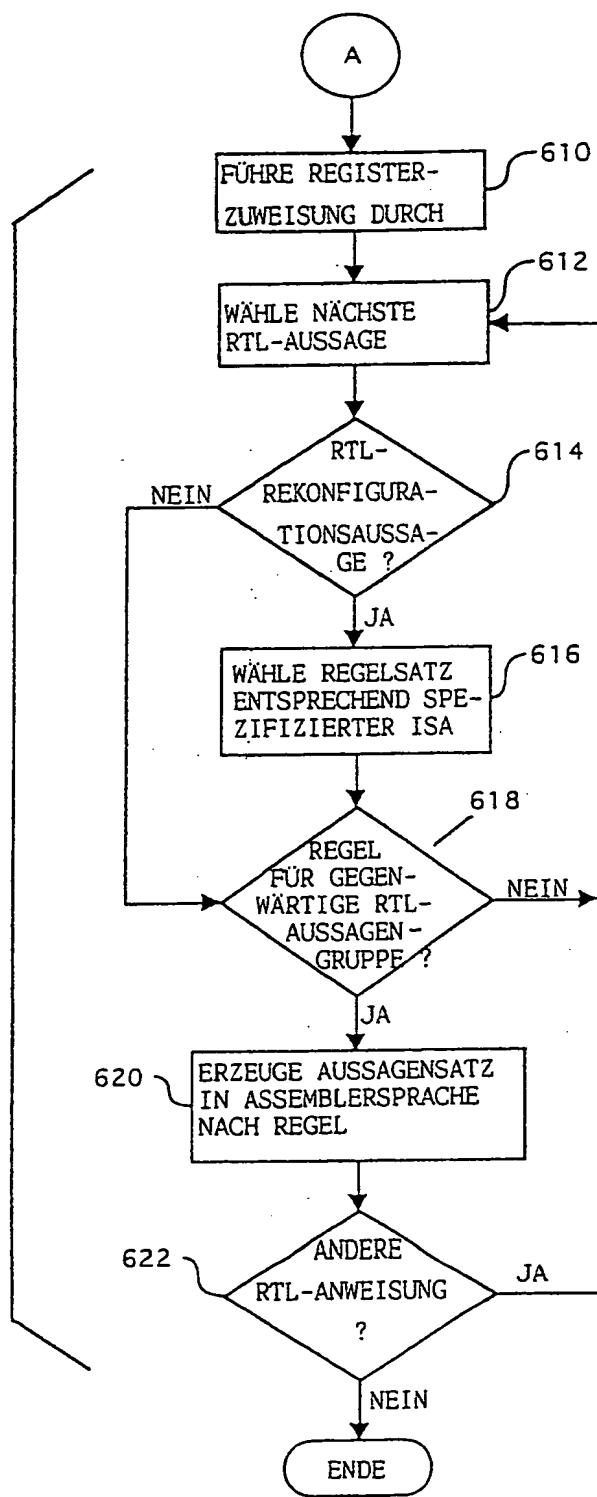


FIG. 3D

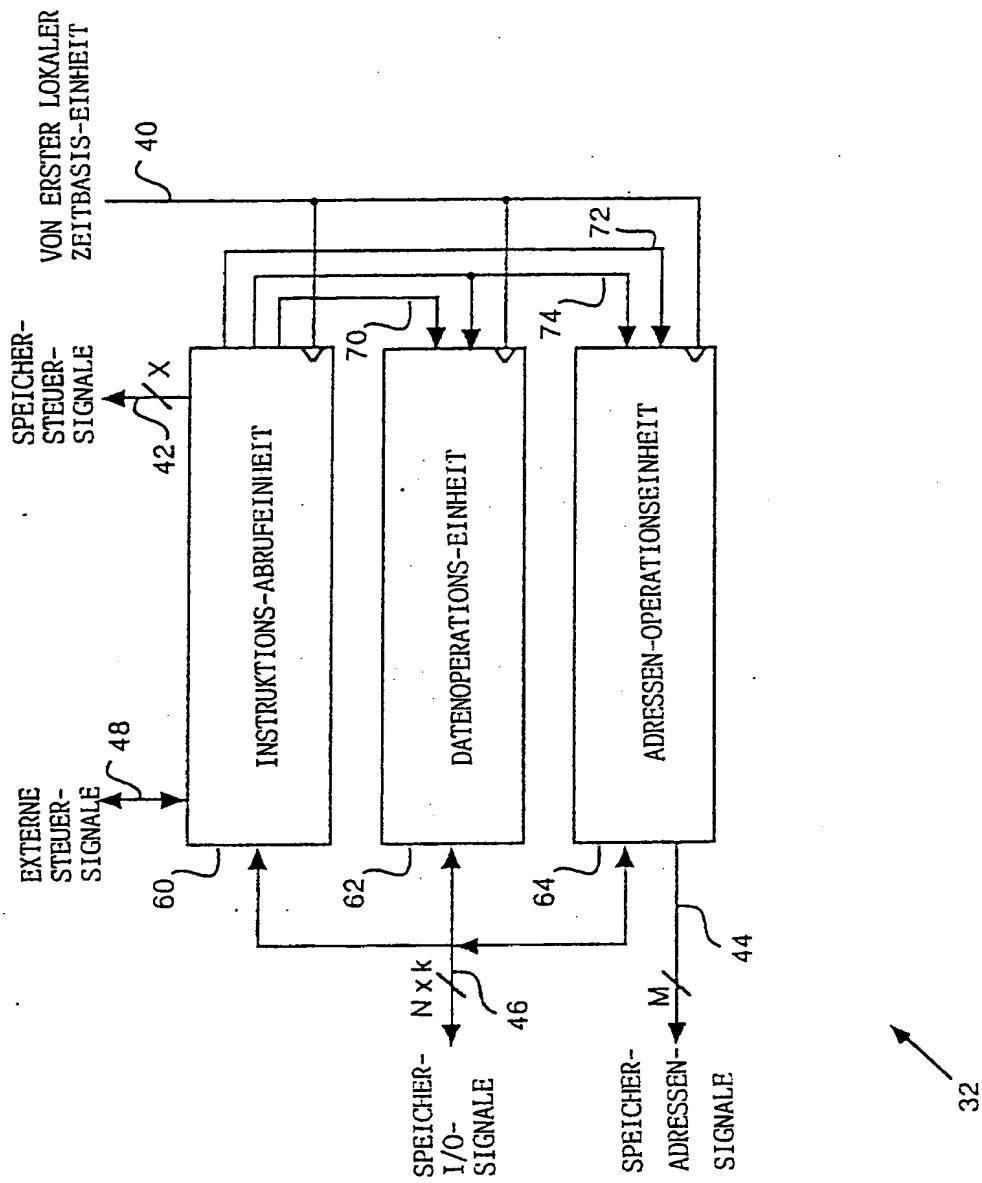
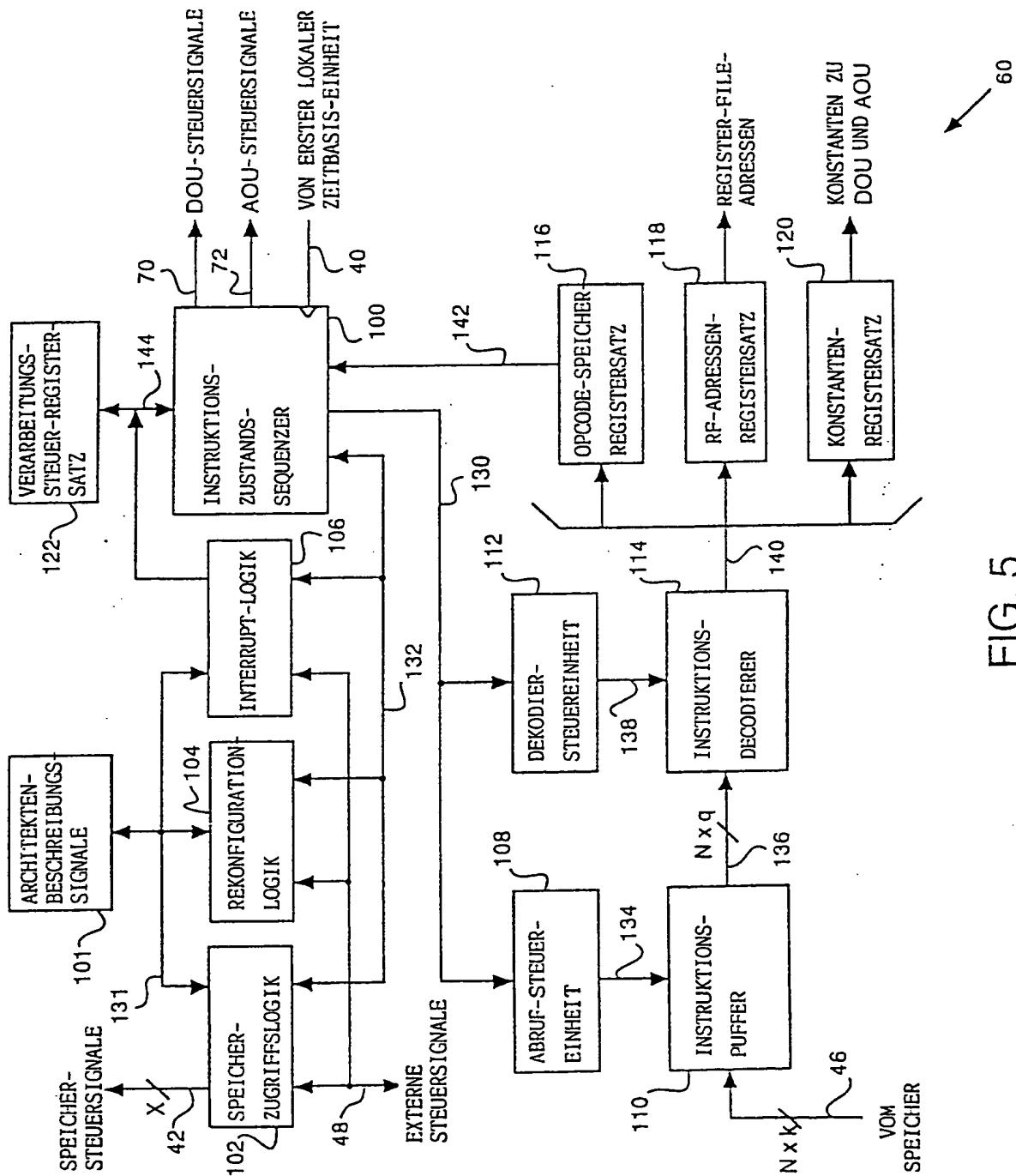


FIG. 4



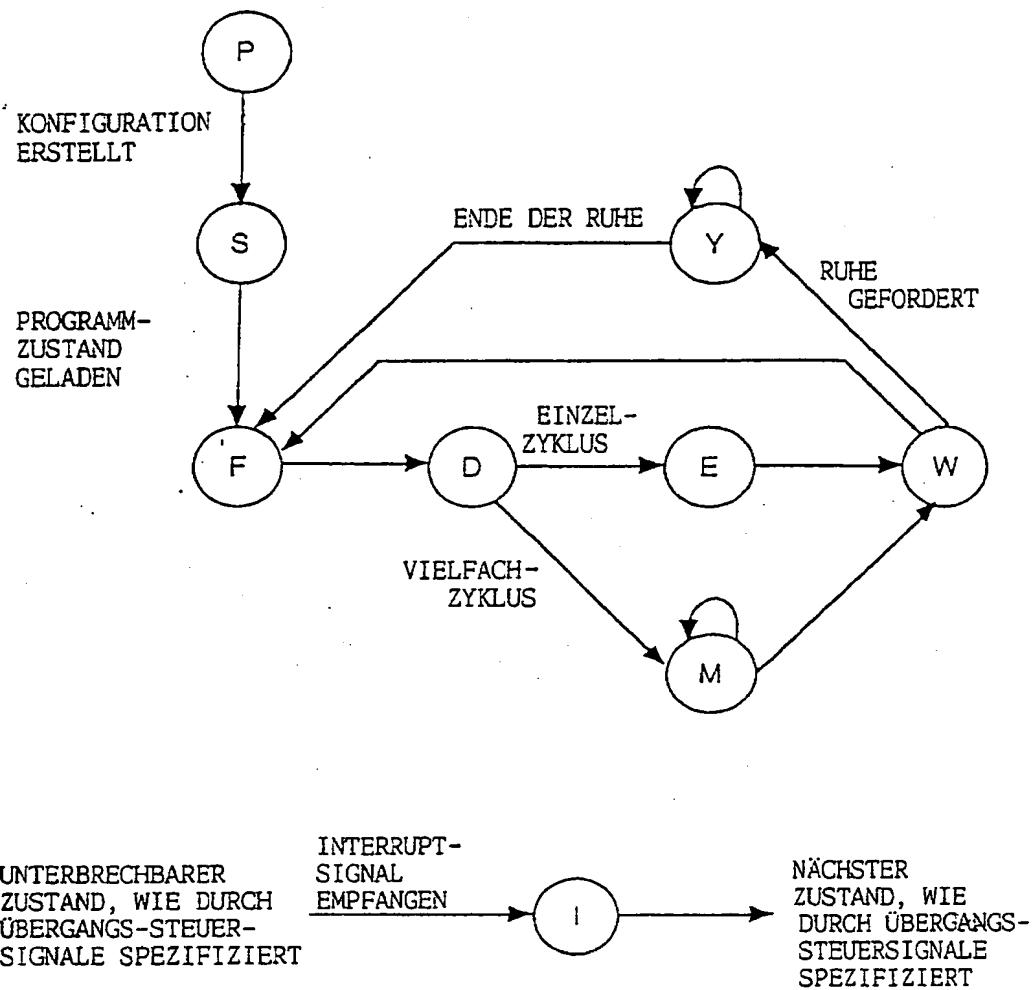


FIG. 6

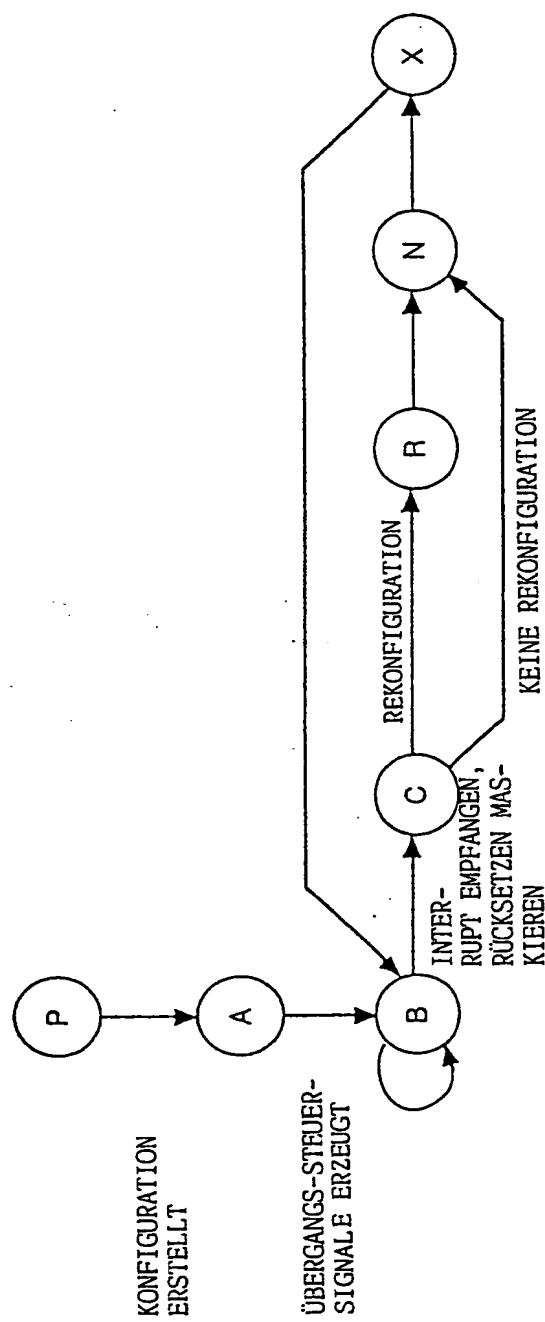
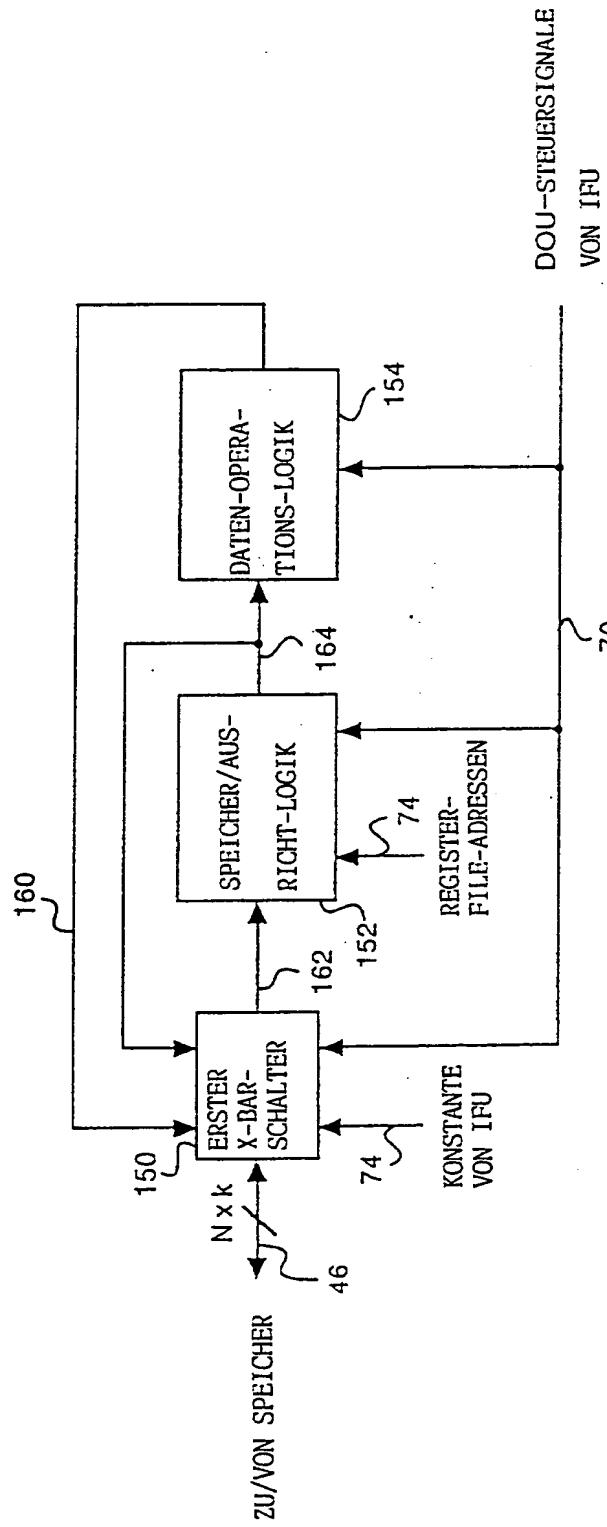


FIG. 7



8
FIG.

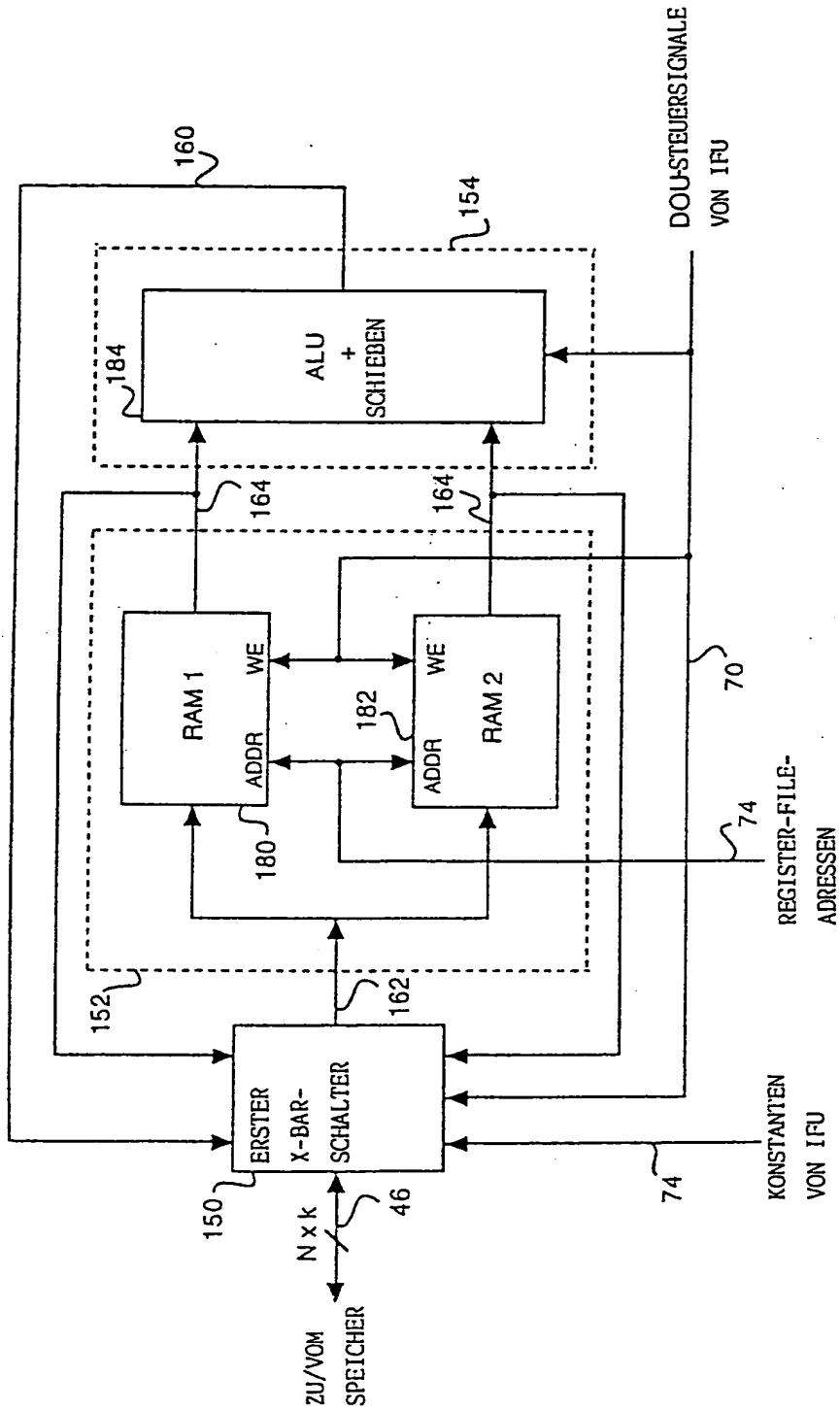


FIG. 9A

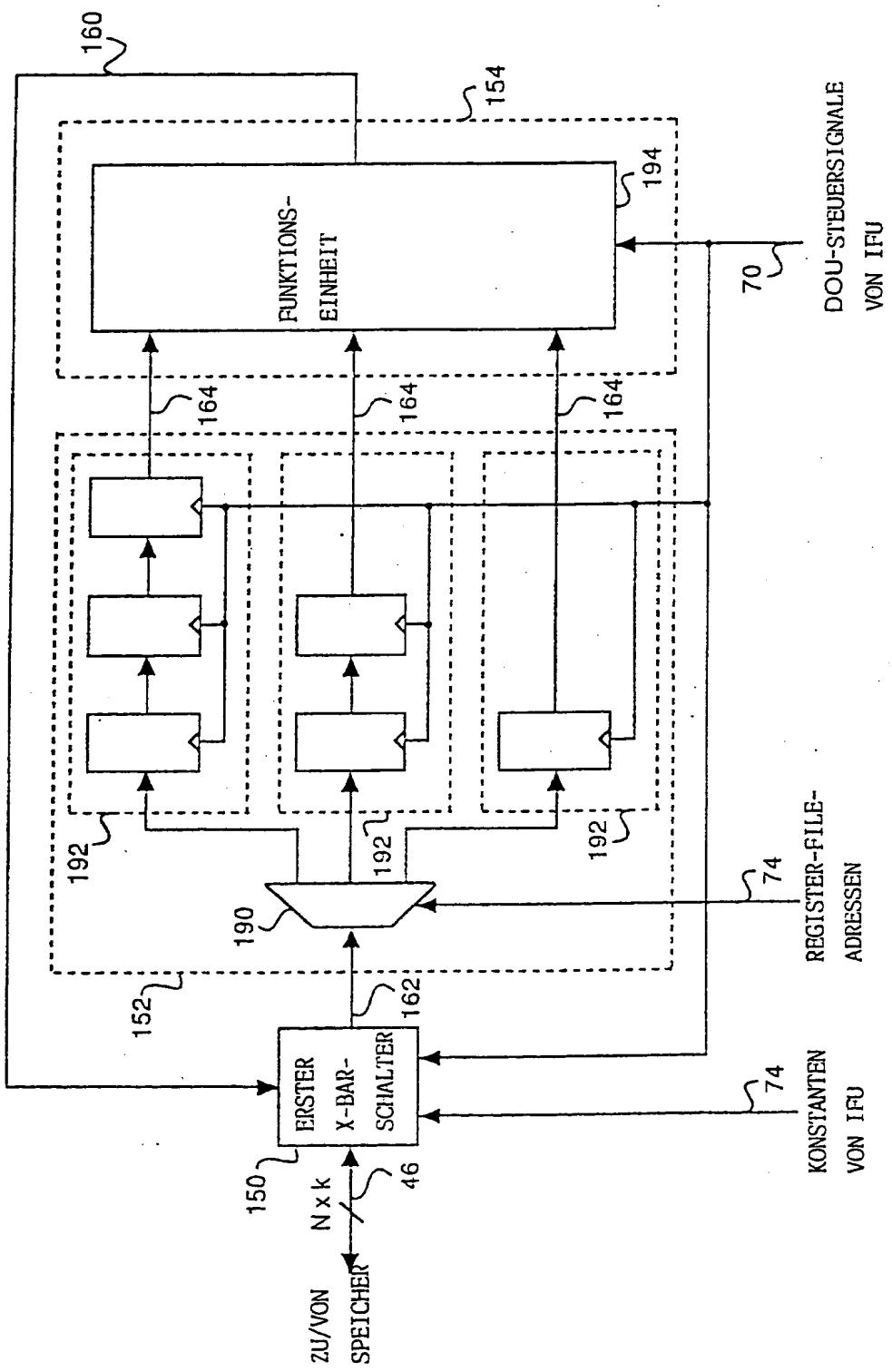


FIG. 9B

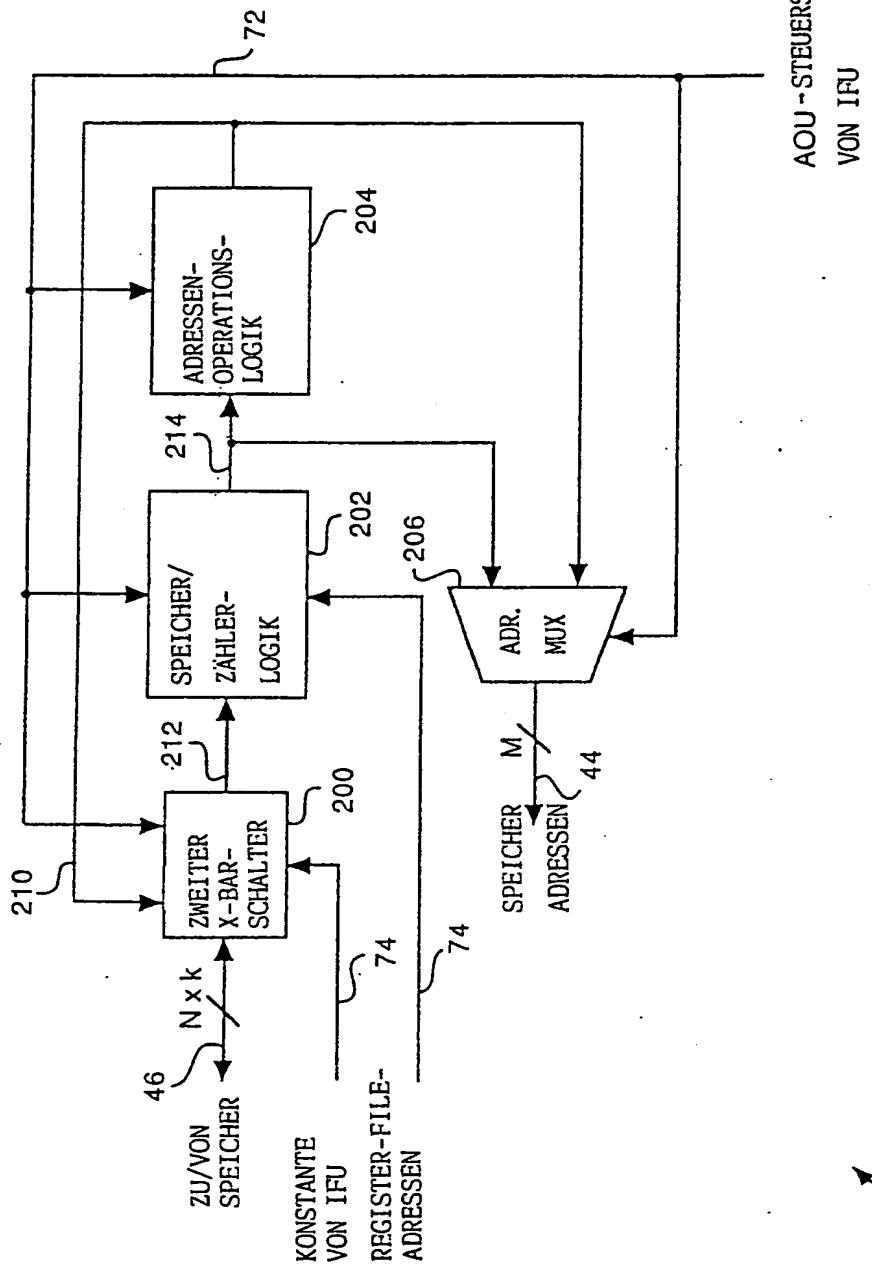


FIG. 10

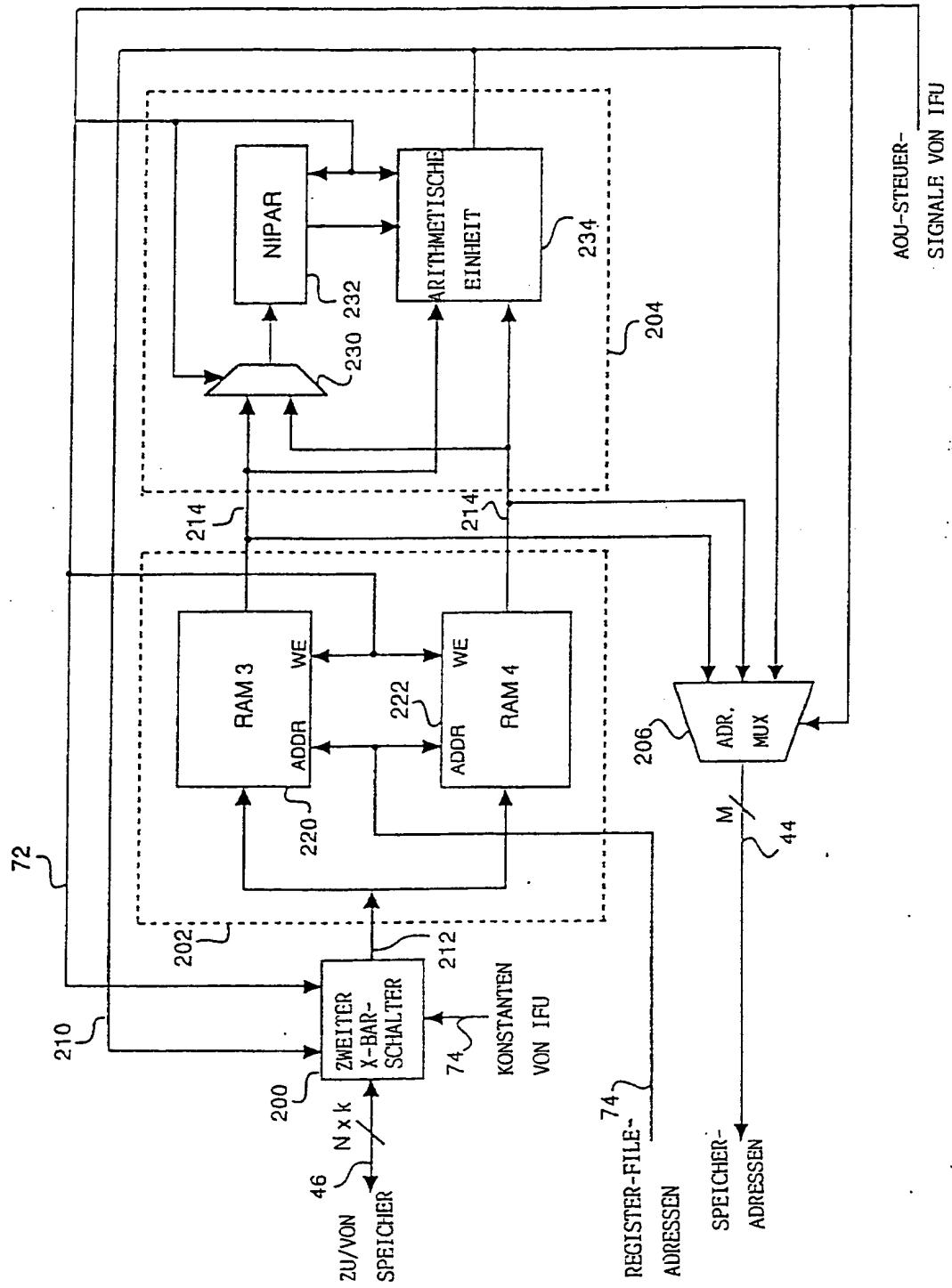


FIG. 11A

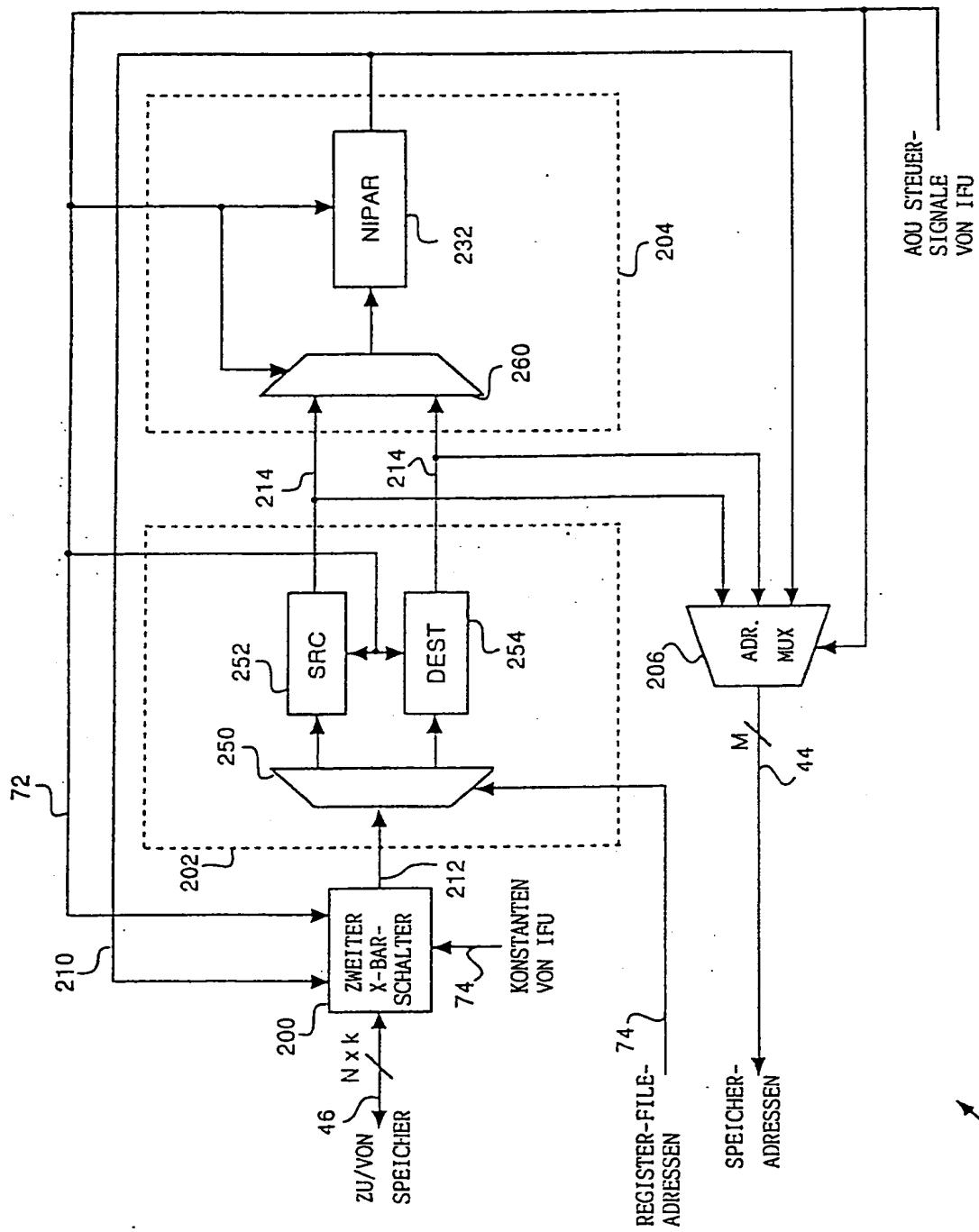


FIG. 11B

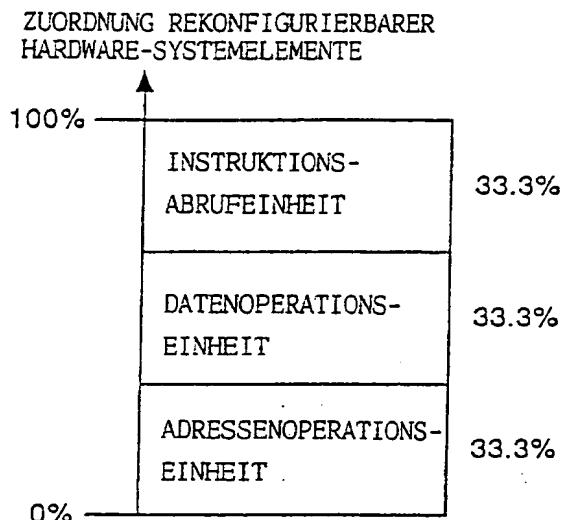


FIG. 12A

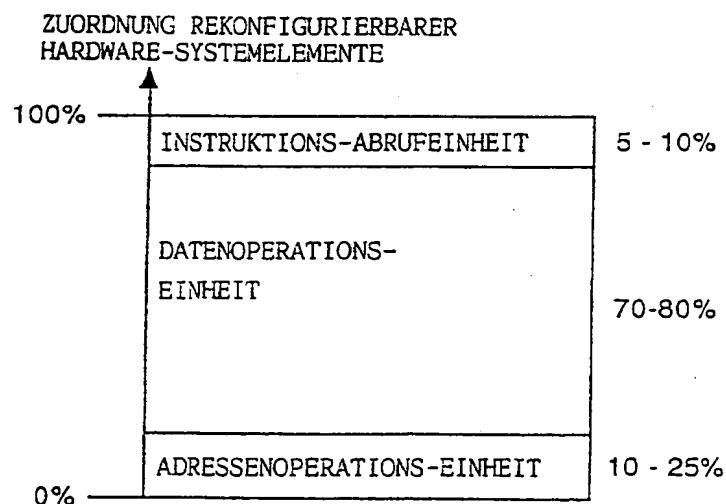


FIG. 12B

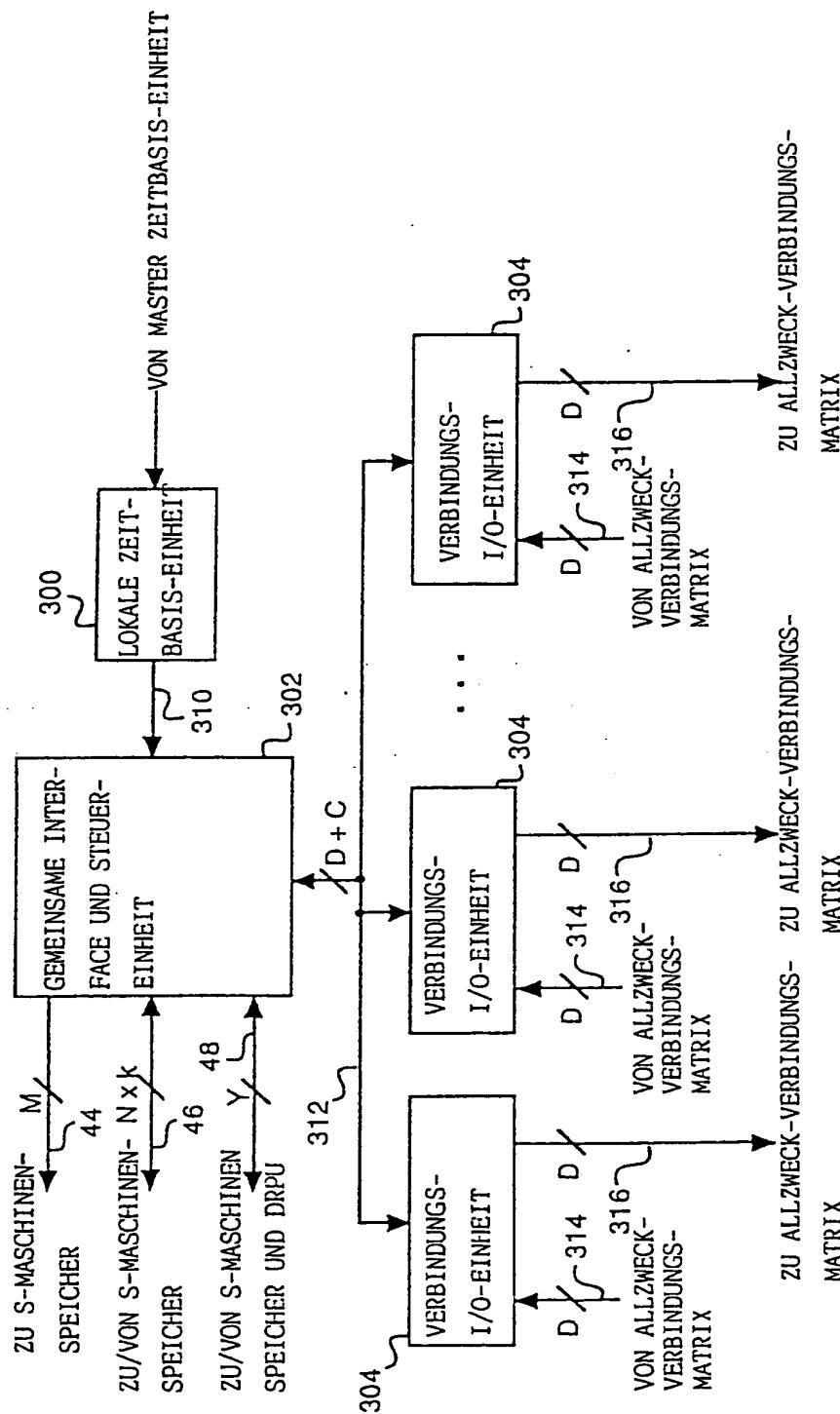


FIG. 13

14

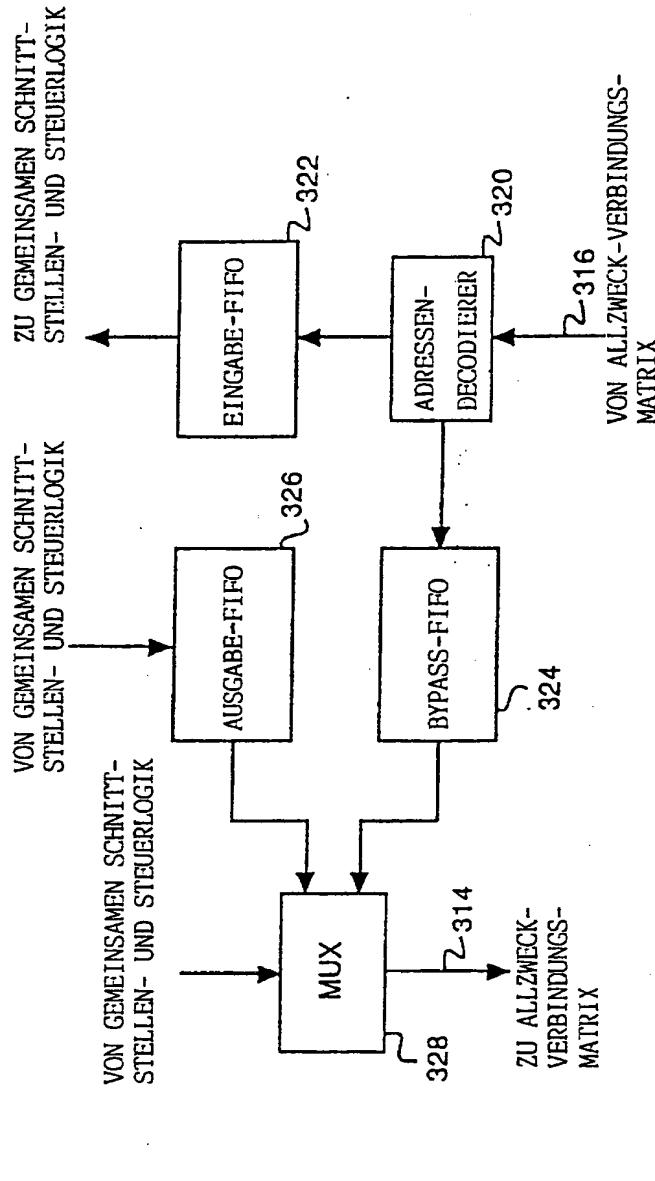
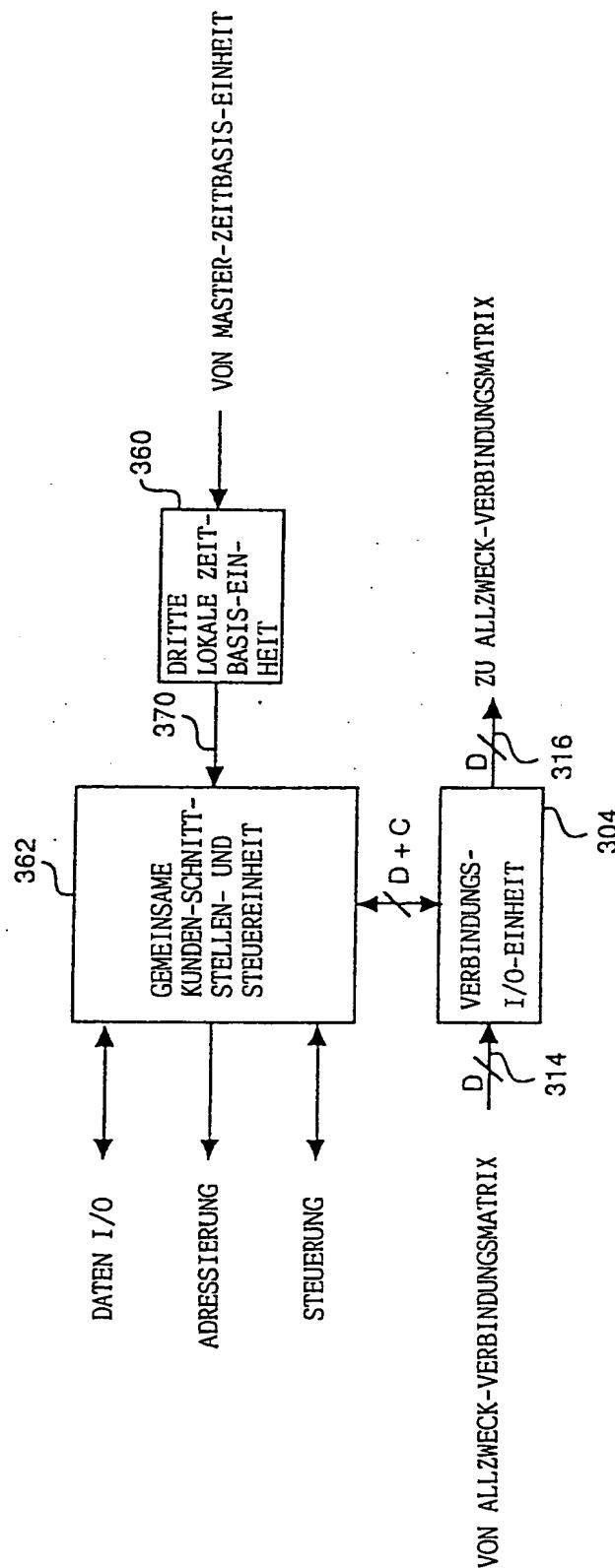


FIG. 14



15
FIG.

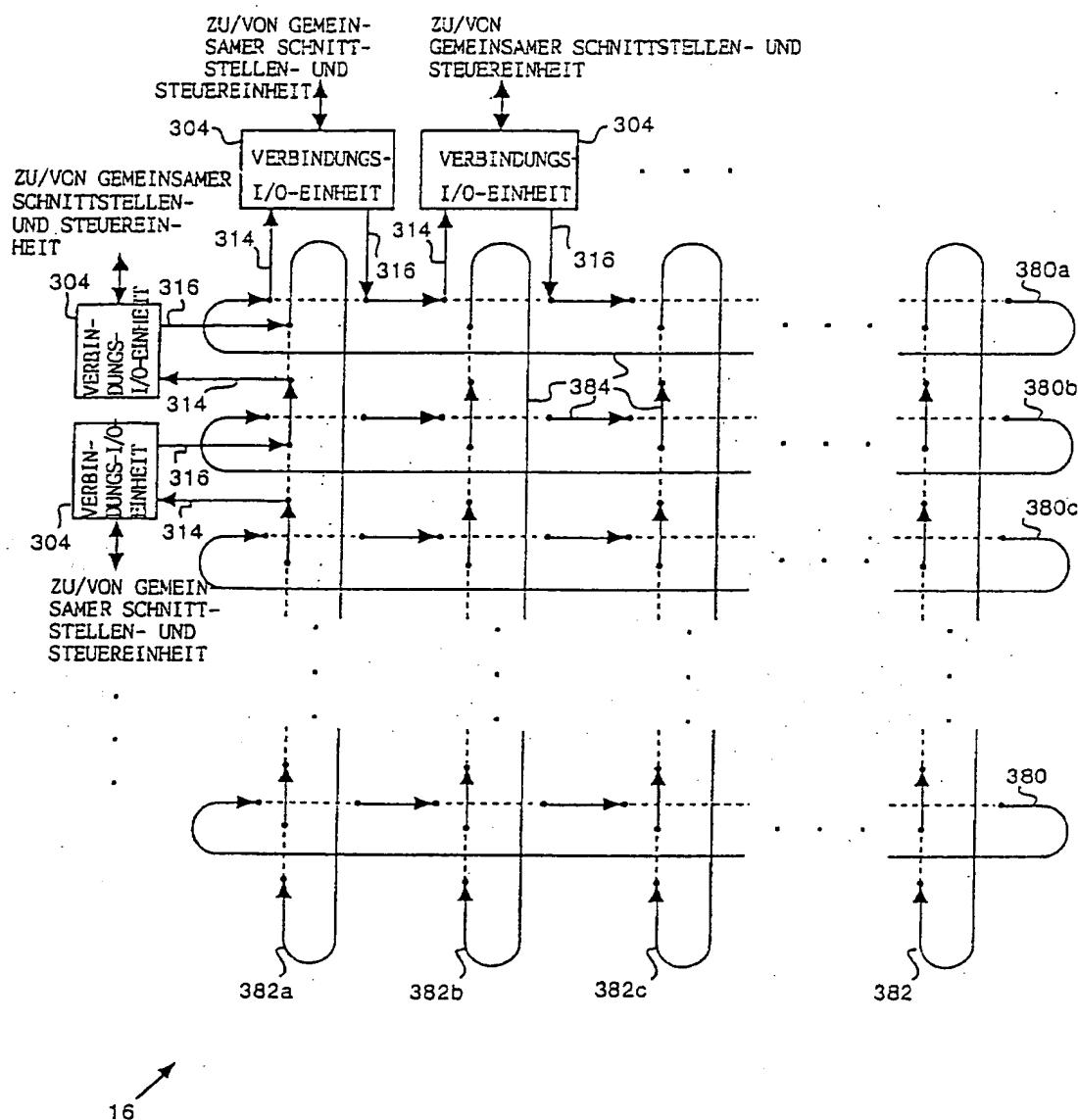
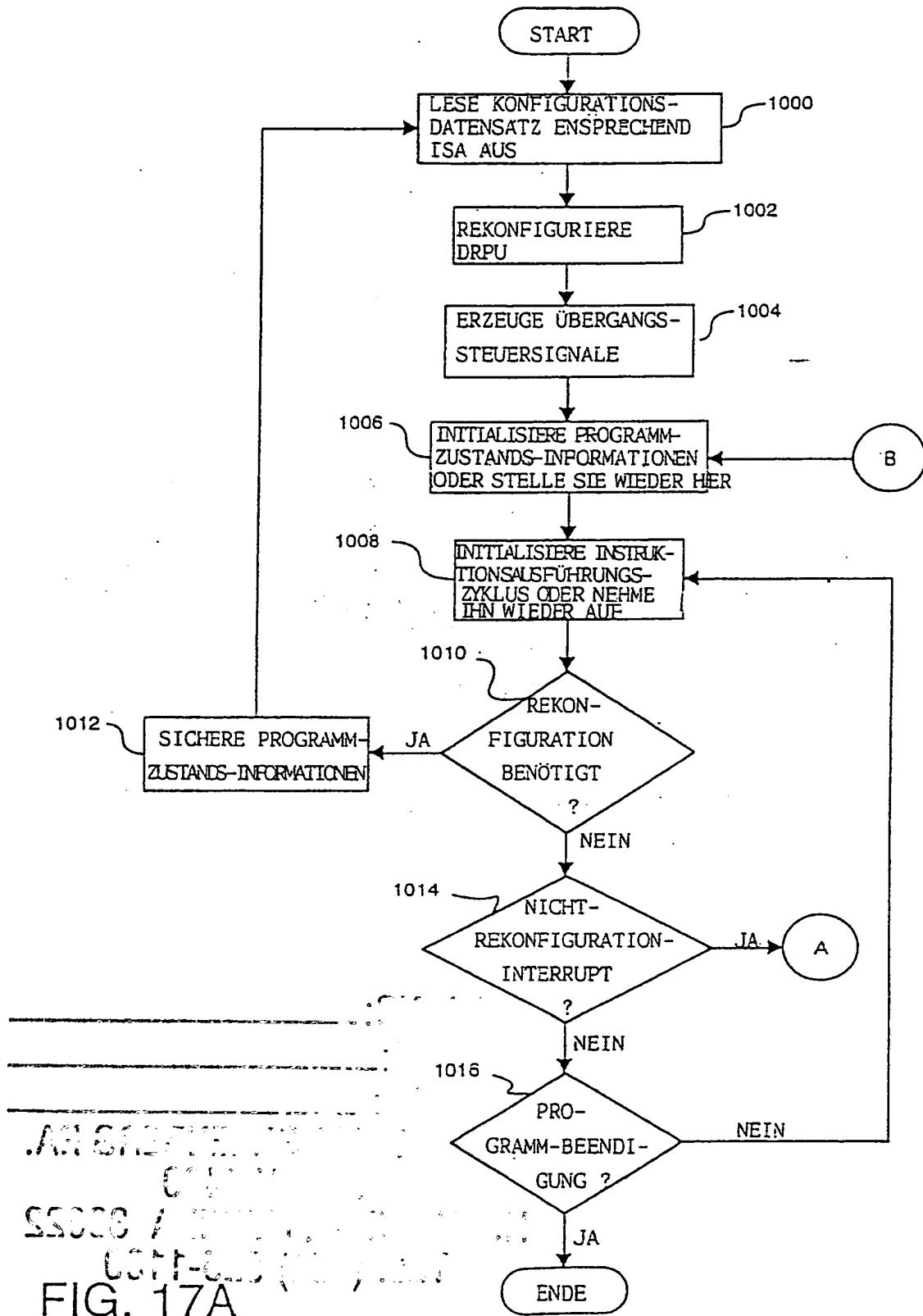


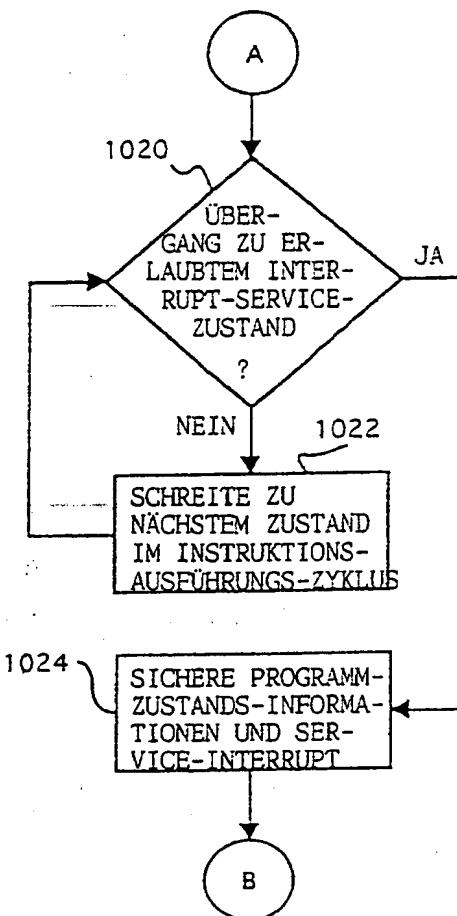
FIG. 16



ANSWER

55000 / 30000

FIG. 17A

DOCKET NO: GR 98 P 8107SERIAL NO: 09/816,926APPLICANT: Arnold et al.

LERNER AND GREENBERG P.A.

P.O. BOX 2480

HOLLYWOOD, FLORIDA 33022

TEL. (954) 925-1100

FIG. 17B